

# COLLEGE OF TECHNOLOGY AND ENGINEERING

MAHARANA PRATAP UNIVERSITY OF AGRICULTURE & TECHNOLOGY  
UDAIPUR (RAJ.)



<b>NAME</b>	<b>HARSH CHANDRAVANSHI</b>
<b>ROLL NO.</b>	<b>27</b>
<b>ENROLLMENT NO.</b>	<b>2019/CTAE/145</b>
<b>SUBJECT CODE</b>	<b>CS235</b>
<b>SUBJECT NAME</b>	<b>OOPS WITH C++</b>
<b>BRANCH</b>	<b>COMPUTER SCIENCE ENGINEERING</b>
<b>YEAR</b>	<b>B.TECH. 2<sup>nd</sup> YEAR</b>

## INDEX

S.NO.	Programs	Date Assigned
1.	Write a Program to design a class having a static member function named showcount () which has the property of displaying the number of objects created of the class.	11/08/20
2.	Write a program using class to process Shopping List for a Departmental Store. The list includes details such as the Code No and Price of each item and performs the operations like Adding, Deleting Items to the list, and Printing the Total value of an Order.	11/08/20
3.	Write a Program which creates & uses array of object of a class. (for e.g. implementing the list of Managers of a Company having details such as Name, Age, etc..).	13/08/20
4.	Write a Program to find Maximum out of Two Numbers using friend function. Note: Here one number is a member of one class and the other number is member of some other class.	13/08/20
5.	Write a Program to swap private data members of classes named as class_1, class_2 using friend function.	18/08/20
6.	Write a Program to design a class complex to represent complex numbers. The complex class should use an external function (use it as a friend function) to add two complex no. the functions should return an object of type complex representing the sum of two complex no.	18/08/20
7.	Write a Program using copy constructor to copy data of an object to another object.	20/08/20
8.	Write a Program to allocate memory dynamically for an object of a given class using class's constructor.	20/08/20
9.	Write a Program to design a class to represent a matrix. The class should have the functionality to insert and retrieve the elements of the matrix.	25/08/20
10.	Write a program to design a class representing complex no. and having the functionality of performing addition & multiplication of two complex numbers using operator overloading	25/08/20
11.	Write a Program to overload operators like *, <<, >> using friend function. The following overloaded operators should work for a class vector.	27/08/20
12.	Write a program for developing a matrix class which can handle integer matrices of different dimensions. Also overload the operator for addition, multiplication & comparison of matrices.	3/09/20
13.	Write a program to overload new / delete operators in a class.	8/12/20
14.	Write a program in C++ to highlight the difference between the overloaded assignment operator and copy constructor.	3/09/20
15.	Write a Program illustrating how the constructors are implemented and the order in which they are called when the classes are	10/09/20

	inherited. Use three classes named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha & beta.	
16.	Write a Program to design a student class representing student roll no. and a test class (derived class of student) representing the scores of the student in various subjects and sports class representing the score in sports. The sports and test class should be inherited by a result class having the functionality to add the scores and display the final result for a student.	8/09/20
17.	Write a program to maintain the records of person with details (Name and Age) and find the eldest among them. The program must use this pointer to return the result. Header file for standard input and output file.	10/09/20
18.	Write a Program to illustrate the use of pointers to objects which are related by inheritance.	15/09/20
19.	Write a program illustrating the use of virtual functions in class.	15/09/20
20.	Write a program to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media ). The class should have the functionality for adding new item, issuing, deposit etc. The program should use the runtime polymorphism.	17/09/20
21.	Write a program to show conversion from string to int and vice-versa.	24/09/20
22.	Write a program showing data conversion between objects of different classes.	29/09/20
23.	Write a program showing data conversion between objects of different classes and conversion routine should reside in destination class.	29/09/20
24.	Write a program implementing basic operation of class ios i.e. setf, unsetf, precision etc.	1/10/20
25.	Write a program to implement I/O operations on characters. I/O operations includes input a string, length of string, store it in a file, fetching the stored characters from it, etc.	6/10/20
26.	Write a program to copy the contents of one file to another.	8/10/20
27.	Write a program to perform read/write binary I/O operation on a file (i.e. write the object of a structure/class to file).	8/10/20
28.	Write a program to maintain a elementary database of employees using files.	13/10/20
29.	Write a program for reading and writing data to and from the file using command line arguments.	15/10/20
30.	Write a program showing implementation of stack class having the functionality of push, pop operations.	20/10/20

31.	Write program to implement a queue class with required operations/ functions	22/10/20
32.	Write a program to implement circular queue class with required operations/ functions.	27/10/20
33.	Write a program implementing linked list as a class. Also Perform some required operations like inserting, deleting nodes & display the contents of entire linked list.	29/10/20
34.	Write a program dynamic memory implementing stack & its operations using memory allocation.	12/11/20
35.	Write a program implementing Queue stack & its operations using dynamic memory allocation.	26/11/20
36.	Write a program to implement the exception handling with multiple catch statements.	8/12/20
37.	Write a program to implement the exception handling with rethrowing in exception.	8/12/20
38.	Write a program to implement the exception handling with the functionality of testing the throw restrictions.	8/12/20

1. Write a Program to design a class having a static member function named showcount () which has the property of displaying the number of objects created of the class.

CODE:

```
//Header files
#include<bits/stdc++.h>
using namespace std;
// Declares a user defined data type test
class test
{
    int code;
    static int count;
//Access specifier
public:
    //setcode function declaration for count
    void setcode(void)
    {
        code = ++count;
    }
    //showcode function declaration to show object code
    void showcode(void)
    {
        cout<<"object number:"<<code<<"\n";
    }
    //showcount function declaration to show count
    static void showcount(void)
    {
        cout<<"count:"<<count<<"\n";
    }
};
int test::count;
int main()
{ //declare object t1,t2 for class test
  test t1, t2;
  //calling function setcode to increase the count
  t1.setcode();
  t2.setcode();
  //calling function showcount
  test::showcount();
  test t3;
  //calling function setcode to increase the count
  t3.setcode();
```

```
//calling function showcount
test::showcount();
//calling function showcode to show code of each object
t1.showcode();
t2.showcode();
t3.showcode();
return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
count:2
count:3
object number:1
object number:2
object number:3
```

2. Write a program using class to process Shopping List for a Departmental Store. The list includes details such as the Code No and Price of each item and performs the operations like Adding, Deleting Items to the list, and Printing the Total value of an Order.

CODE:

```
//Header files
#include<bits/stdc++.h>
using namespace std;
//declaring const m for number of items
const int m=50;
class ITEMS
{ //declaring variable itemCode,itemPrice and count
  int itemCode[m];
  float itemPrice[m];
  int count;
// Access specifier
public:
  //declaring/define CNT function for count
  void CNT(void) {
    count=0;
  }
  //declaring/define all function for different operations
  void getitem(void);
  void displaySum(void);
  void remove(void);
  void displayItems(void);
};
//Function to enter items in the shopping list
void ITEMS::getitem(void)
{

  cout<<"Enter item code:"<<endl;
  std::cin>>itemCode[count];
  cout<<"Enter Item cost"<<endl;
  cin>>itemPrice[count];
  count++;
}
//Function to display total amount of items
void ITEMS::displaySum(void)
{
```

```

float sum=0;
for (int i=0;i<count;i++)
    sum=sum+itemPrice[i];
cout<<"\nTotal Value:"<<sum<<"\n";
}
//Function to delete items
void ITEMS::remove(void)
{

int a;
cout<<"Enter Item Code:"<<endl;
cin>>a;
for (int i=0;i<count;i++)
    if (itemCode[i] == a)
        itemPrice[i]=0;
}
//Function to display items
void ITEMS::displayItems(void)
{

cout<<"\nCode Price\n";
for (int i=0;i<count;i++)
{
    cout<<"\n"<<itemCode[i];
    cout<<" "<<itemPrice[i];
}
cout<<"\n";
}
int main()
{ // Declares a user defined data type ITEMS
ITEMS order;
order.CNT();
int x;
do
{
    cout<<"\nYou can do the following:"<<" "
        <<"Enter appropriate number\n";
    cout<<"\n1 : Add an Item";
    cout<<"\n2 : Display Total Value";
    cout<<"\n3 : Delete an Item";
    cout<<"\n4 : Display all items";
}

```



```
cout<<"\n5 : Quit";
cout<<"\n\nWhat is your option?"<<endl;
cin>>x;
switch (x)
{
    //calling function to add items
case 1: order.getitem();
    break;
    //calling function to display total value of items
case 2: order.displaySum();
    break;
    //calling function to delete item
case 3: order.remove();
    break;
    //calling function to display items
case 4: order.displayItems();
    break;
default: cout<<"Error in input"<<endl;
}
} while (x!=5);
return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
```

```
You can do the following: Enter appropriate number
```

```
1 : Add an Item
2 : Display Total Value
3 : Delete an Item
4 : Display all items
5 : Quit
```

```
What is your option?
```

```
1
Enter item code:
1
Enter Item cost
50
```

```
You can do the following: Enter appropriate number
```

```
1 : Add an Item
2 : Display Total Value
3 : Delete an Item
4 : Display all items
5 : Quit
```

```
What is your option?
```

```
2
Total Value:50
```

```
You can do the following: Enter appropriate number
```

```
1 : Add an Item
2 : Display Total Value
3 : Delete an Item
4 : Display all items
5 : Quit
```

```
What is your option?
```

```
4
Code Price
```

```
1 50
```

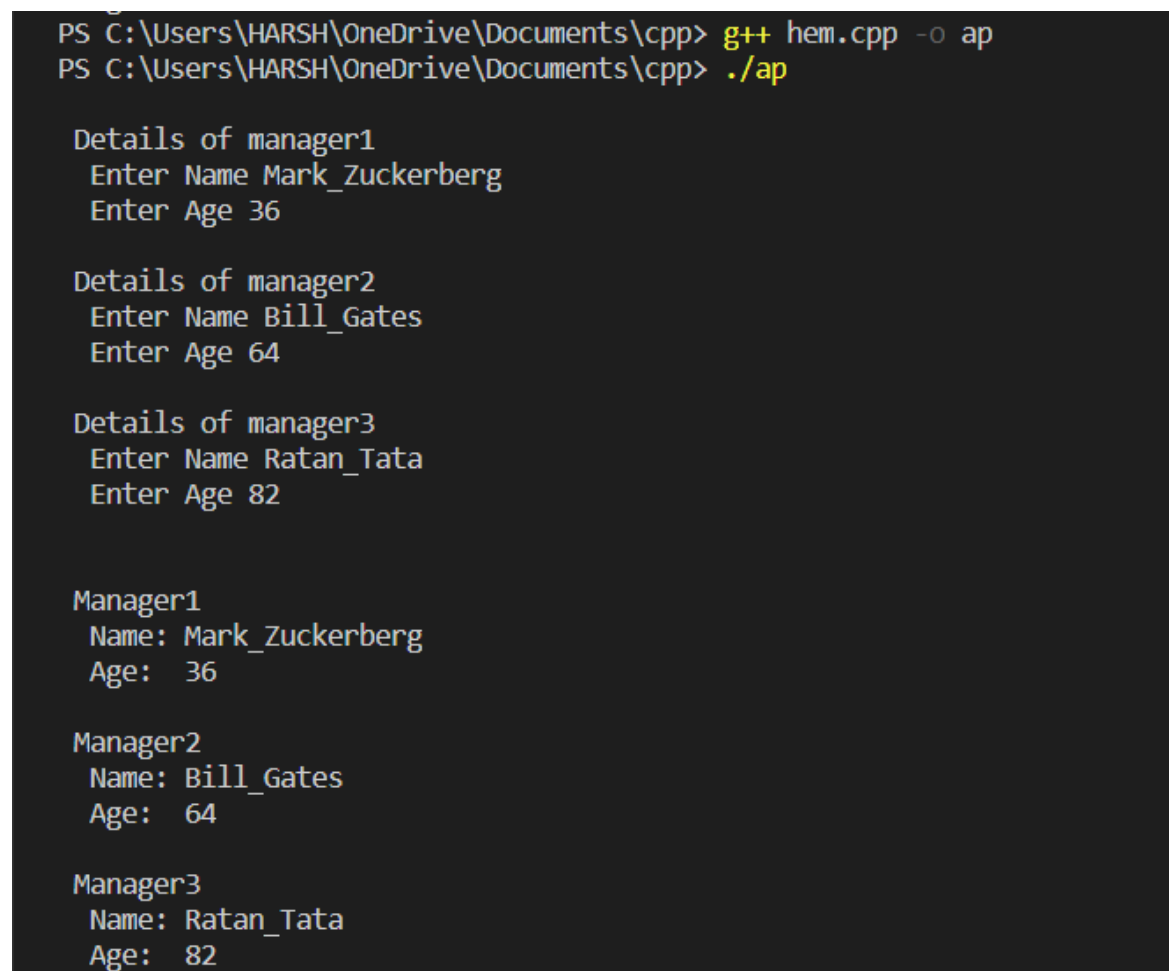
3. Write a Program which creates & uses array of object of a class. (for e.g. implementing the list of Managers of a Company having details such as Name, Age, etc..).

CODE:

```
//Header Files
#include <bits/stdc++.h>
using namespace std;
// Declares a user defined data type employee
class employee
{ //Declare name and age variable
  char name[30];
  float age;
public:
  //Define function getdata and putdata for operations
  void getdata(void);
  void putdata(void);
};
//Function to get values from user
void employee::getdata(void)
{
  cout<<" Enter Name"<<" ";
  cin>>name;
  cout<<" Enter Age"<<" ";
  cin>>age;
}
//Function to show entered value
void employee::putdata(void)
{
  cout<<" Name:"<<" "<<name<<"\n";
  cout<<" Age: "<<" "<<age<<"\n";
}
//Assign a const size to take value of only three manager
const int size=3;
int main()
{ //Define array of object size for class employee
  employee manager[size];
  for (int i=0; i<size; i++)
  {
    cout<<"\n Details of manager"<<i+1<<"\n";
    //calling function getdata to get value from user
    manager[i].getdata();
  }
}
```

```
cout<<"\n";
for (int i=0; i<size; i++)
{
    cout<<"\n Manager"<<i+1<<"\n";
    //calling function to show output entered by user
    manager[i].putdata();
}
return 0;
}
```

OUTPUT:



```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

Details of manager1
Enter Name Mark_Zuckerberg
Enter Age 36

Details of manager2
Enter Name Bill_Gates
Enter Age 64

Details of manager3
Enter Name Ratan_Tata
Enter Age 82

Manager1
Name: Mark_Zuckerberg
Age: 36

Manager2
Name: Bill_Gates
Age: 64

Manager3
Name: Ratan_Tata
Age: 82
```

4. Write a Program to find Maximum out of Two Numbers using friend function.

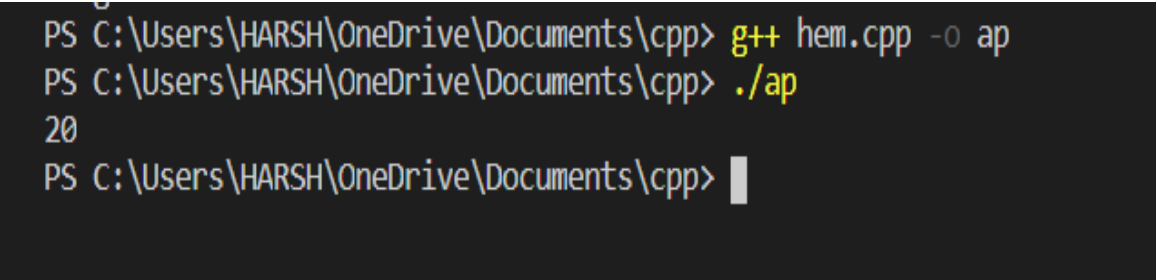
Note: Here one number is a member of one class and the other number is member of some other class.

CODE:

```
//Header File
#include <bits/stdc++.h>
using namespace std;
//Declares a user defined data type ABC
class ABC;
//Declares a user defined data type XYZ
class XYZ
{
    int x;
public:
    //Function to take value and set it to variable
    void setvalue(int i)
    {
        x=i;
    }
    friend void max(XYZ, ABC);
};
class ABC
{
    int a;
public:
    //Function to take value and set it to variable
    void setvalue(int i)
    {
        a=i;
    }
    friend void max(XYZ, ABC);
};
//Define function max
void max(XYZ m, ABC n)
{
    if (m.x>=n.a)
        cout<<m.x;
    else
        cout<<n.a;
}
int main()
```

```
{ //Declare object abc for class abc
  ABC abc;
  //Calling function to take value
  abc.setvalue(10);
  //Declare object abc for class xyz
  XYZ xyz;
  //Calling function to take value
  xyz.setvalue(20);
  //Calling function to pass value to function max
  max(xyz, abc);
  return 0;
}
```

OUTPUT:



```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
20
PS C:\Users\HARSH\OneDrive\Documents\cpp> |
```

5. Write a Program to swap private data members of classes named as class\_1, class\_2 using friend function.

CODE:

```
//Header File mainly for standard input output
#include <iostream>
#include <conio.h>
using namespace std;
//Define class_2 to use in class_1
class class_2;
//Declaration of class_1
class class_1
{ //value1 variable defined in private member
    int value1;
    //access specifier
public:
    //indata function declaration to take value_1 from user
    void indata(int a)
    {
        value1 = a;
    }
    //display function declaration to show inputted value_1 to user
    void display(void)
    {
        cout << value1 << "\n";
    }
    //Friend function declaration to use private members variable of class

    friend void exchange(class_1 &, class_2 &);
};
//Declaration of class_2
class class_2
{ //value2 variable defined in private member
    int value2;
    //access specifier
public:
    //indata function declaration to take value_2 from user
    void indata(int a)
    {
        value2 = a;
    }
}
```

```

//display function declaration to show inputted value_1 to user
void display(void)
{
    cout << value2 << "\n";
}
//Friend function declaration to use private members variable of class
friend void exchange(class_1 &, class_2 &);
};
//Exchange function declaration with prior way to swap value1 to value2
void exchange(class_1 &x, class_2 &y)
{
    int temp = x.value1;
    x.value1 = y.value2;
    y.value2 =
        temp;
}
int main()
{ //object C1 declaration of class_1

    class_1 C1;
    //object C2 declaration of class_2
    class_2 C2;
    //input value to class_1
    C1.indata(100);
    //input value to class_2
    C2.indata(200);
    cout << "Values before exchange"
        << "\n";
    //function call of class_1 to display value before swapping
    C1.display();
    //function call of class_2 to display value before swapping
    C2.display();
    //exchange function call to swap values
    exchange(C1, C2);
    cout << "Values after exchange"
        << "\n";
    //function call of class_1 to display value after swapping
    C1.display();
    //function call of class_2 to display value after swapping
    C2.display();
    return 0;
}

```



```
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Values before exchange
100
200
Values after exchange
200
100
```

6. Write a Program to design a class complex to represent complex numbers. The complex class should use an external function (use it as a friend function) to add two complex no. the functions should return an object of type complex representing the sum of two complex no.

CODE:

```
//Header File mainly for standard input output
#include <iostream>
#include <conio.h>
using namespace std;
//Complex class definition
class complex
{ //variable x,y declaration under private member
    float x;
    float y;
    //Access Specifier
public:
    //Input function definition to take real and imaginary value of complex number from user
    void input(float real, float img)
    {
        x = real;
        y = img;
    }
    //Friend function declaration to use x, y private member variable
    friend complex sum(complex, complex);
    //Show function declaration with complex data type
    void show(complex);
};
//sum function definition to do the sum
complex sum(complex c1, complex c2)
{ //variable C3 declaration with complex data type to store sum of two complex number
    complex c3;
    //Sum of real value of two complex number
    c3.x = c1.x + c2.x;
    //Sum of imaginary value of two complex number
    c3.y = c1.y + c2.y;
    //returning value c3
    return (c3);
}
//show function definition to show complex value in following format
void complex ::show(complex c)
```

```

{
    cout << c.x << "+j" << c.y << "\n";
}
int main()
{ //object A,B,C declaration of class complex
    complex A, B, C;
    //inputting data through object A
    A.input(3.1, 5.65);
    //inputting data through object B
    B.input(2.75, 1.2);
    //Calling function sum and store the value of sum in object C
    C = sum(A, B);
    cout << "A=";
    //Calling function show to print value in object A
    A.show(A);
    cout << "B=";
    //Calling function show to print value in object B
    B.show(B);
    cout << "C=";
    //Calling function show to print value in object C
    C.show(C);
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
A=3.1+j5.65
B=2.75+j1.2
C=5.85+j6.85

```

7. Write a Program using copy constructor to copy data of an object to another object.

CODE:

```
//Header file for standard input and output and necessary files
#include <iostream>
#include <conio.h>
using namespace std;
//Code Class definition
class code
{ //id variable defined in default private class
  int id;
  //Access specifier
public:
  //default constructor code declaration
  code() {}
  //Paramaterised constructor code definition with int 'a' as an argument
  code(int a)
  {
    id = a;
  }
  //constructor code definition with code '&x' as an argument
  code(code &x)
  {
    id = x.id;
  }
  //function display to show the value of id
  void display(void)
  {
    cout << id;
  }
};
int main()
{ //Constructor call
  code A(100);
  code B(A)
  code C = A;
  code D;
  //initialise D with A
  D = A;
  //Displaying value of ids of all four objects
  cout << "\n id of A:";
  A.display();
  cout << "\n id of B:";
  B.display();
  cout << "\n id of C:";
  C.display();
```

```
cout << "\n id of D:";  
D.display();  
return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
  
id of A:100  
id of B:100  
id of C:100  
id of D:100
```

8. Write a Program to allocate memory dynamically for an object of a given class using class's constructor.

CODE:

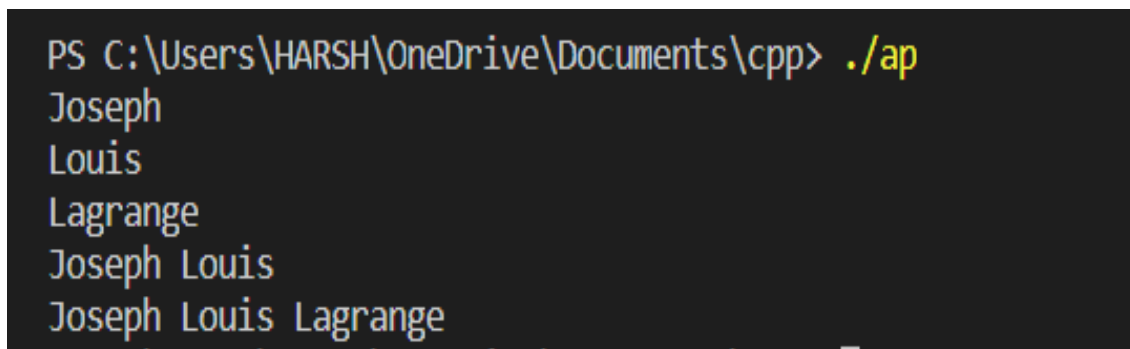
```
//Header file for standard input and output and necessary files
#include <iostream>
#include <conio.h>
//Header file to include all string properties
#include <string.h>
using namespace std;
//Defining Class String
class String
{
    //Variable defined in default private member
    char *name;
    int length;
    //access specifier
public:
    //String constructor definition with no argument
    String()
    {
        length = 0;
        name = new char[length + 1];
    }
    //String constructor definition with char pointer as argument
    String(char *s)
    {
        //strlen "stl" to calculate length of string
        length = strlen(s);
        name = new char[length + 1];
        //strcpy "stl" to copy s value to name
        strcpy(name, s);
    }
    //display function to show value of name char pointer
    void display(void)
    {
        cout << name << "\n";
    }
    //join function declaration
    void join(String &a, String &b);
};
//definition of function join using constructor string
void String ::join(String &a, String &b)
{
    length = a.length + b.length;
    delete name;
```

```

    name = new char[length + 1];
    strcpy(name, a.name);
    strcat(name, b.name);
};
int main()
{
    //initialisation of first pointer
    char *first = "Joseph ";
    //initialisation of string class objects
    String name1(first), name2("Louis "), name3("Lagrange"), s1, s2;
    //calling join function with string class object s1 and s2
    s1.join(name1, name2);
    s2.join(s1, name3);
    //calling display function to show values using string different objects
    name1.display();
    name2.display();
    name3.display();
    s1.display();
    s2.display();
    return 0;
}

```

OUTPUT:



```

PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Joseph
Louis
Lagrange
Joseph Louis
Joseph Louis Lagrange

```

9. Write a Program to design a class to represent a matrix. The class should have the functionality to insert and retrieve the elements of the matrix.

CODE:

```
//Header file to include standard or user-defined files
#include <iostream>
//namespace to use same name variable in different namespace and scopes
using namespace std;
//Matrix class definition
class matrix
{ //default private member
    //pointer of pointer p
    int **p;
    //d1 and d2 variable
    int d1, d2;

    //access specifier
public:
    //declaration of paramatarized constructor with two int type argument x and y
    matrix(int x, int y);
    //get_element function definition to take values from user
    void get_element(int i, int j, int value)
    {
        p[i][j] = value;
    }
    //put_element function definition to show desired output
    int put_element(int i, int j)
    {
        return p[i][j];
    }
};
//definition of matrix constructor
matrix ::matrix(int x, int y)
{ //assign value of x in d1
    d1 = x;
    //assign value of y in d2
    d2 = y;
    p = new int *[d1];
    for (int i = 0; i < d1; i++)
        p[i] = new int[d2];
}
int main()
```



```

{ //m and n variable
  int m, n;
  cout << "Enter size of matrix" << endl;
  //Enter value greater than 2 as we use 1 and 2 index in out output function call
  cin >> m >> n;
  //implicit call of matrix constructor
  matrix A(m, n);
  cout << "Enter Matrix Element row by row:" << endl;
  int i, j, value;
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      {
        cin >> value;
        //called get_element function to get input from user
        A.get_element(i, j, value);
      }
  cout << "\n";
  //called put_element function to show output
  cout << A.put_element(1, 2);
  return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Enter size of matrix
3
3
Enter Matrix Element row by row:
1
2
3
4
5
6
7
8
9
6

```

10. Write a program to design a class representing complex no. and having the functionality of performing addition & multiplication of two complex numbers using operator overloading.

CODE:

```
//Header file to include standard or user-defined files
#include <iostream>

//namespace to use same name variable in different namespace and scopes
using namespace std;

//complex class definition
class complex
{
    //private access specifier
private:
    //definition of float variable real and imag
    float real, imag;
    //public access specifier
public:
    //default complex constructor
    complex()
    {
    }
    //Parameterized Constructors with argument of float type r and i
    complex(float r, float i)
    { //assign value of r in real
        real = r;
        //assign value of i in imag
        imag = i;
    }
    //getdata function definition to get real and imaginary values from user
    void getdata()
    {
        float r, i;
```

```

cout << endl
    << "Enter real and imaginary part ";
cin >> r >> i;
real = r;
imag = i;
}
//alternative way of passing value, setdata function definition to set r and i value to real
and imag respectively
void setdata(float r, float i)
{
    //assign value of r in real
    real = r;
    //assign value of i in imag
    imag = i;
}
//displaydata function definition to display required output
void displaydata()
{

    cout << endl
        << "real = " << real;
    cout << endl
        << "Imaginary = " << imag << "\n";
}
//'+ operator overloading
complex operator+(complex c)
{
    complex t;
    //add real value of complex number
    t.real = real + c.real;
    //add imaginary value of complex number
    t.imag = imag + c.imag;
}

```

```

    return t;
}
/**' operator overloading
complex operator*(complex c)
{
    complex t;
    //multiplying real value of complex number
    t.real = real * c.real - imag * c.imag;
    //multiplying imaginary value of complex number
    t.imag = real * c.imag + c.real * imag;
    return t;
}
};
int main()
{ //c1,c3,c4 object declaration and implicit constructor call using c2 object
    complex c1, c2(1.2, -2.5), c3, c4;
    //implicit setdata constructor call using c1 object
    c1.setdata(2.0, 2.0);
    //adding c1 and c2 using overloaded '+' operator and assign them to object c3
    c3 = c1 + c2;
    cout << "\n"
        << "c3 = c1 + c2"
        << "\n";
    //displaydata function call to show output using c1,c2,c3 object
    c1.displaydata();
    // real = 2
    // Imaginary = 2
    c2.displaydata();
    // real = 1.2
    // Imaginary = -2.5
    c3.displaydata();
    // real = 3.2

```

```

// Imaginary = -0.5
//getdata function call using c4 object
c4.getdata();
//implicit call to complex constructor using c5 object and c6 object declaration
complex c5(2.5, 3.0), c6;
//multiply c4 and c5 using overloaded '*' operator and assign them to object c6
c6 = c4 * c5;
cout << "\n"
    << "c6 = c4 * c5"
    << "\n";
//display data function call using c4,c5,c6 object
//value given by user
c4.displaydata();
// real = 1
// Imaginary = 1
c5.displaydata();
// real = 2.5
// Imaginary = 3
c6.displaydata();
// real = -0.5
// Imaginary = 5.5
//c7 object declaration
complex c7;
//both adding and multiplying using both '+' and '*' overloaded operator and assign them
to c7 object
c7 = c1 + c2 * c3;
cout << "\n"
    << "c7 = c1 + c2 * c3"
    << "\n";
//displaydata function call using c1,c2,c3, and c7 objects
c1.displaydata();
// real = 2

```

```

// Imaginary = 2
c2.displaydata();
// real = 1.2
// Imaginary = -2.5
c3.displaydata();
// real = 3.2
// Imaginary = -0.5
c7.displaydata();
// real = 4.59
// Imaginary = -6.6
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

c3 = c1 + c2
real = 2
Imaginary = 2

real = 1.2
Imaginary = -2.5

real = 3.2
Imaginary = -0.5

Enter real and imaginary part 1
1

c6 = c4 * c5
real = 1
Imaginary = 1

real = 2.5
Imaginary = 3

real = -0.5
Imaginary = 5.5

c7 = c1 + c2 *c3
real = 2
Imaginary = 2

real = 1.2
Imaginary = -2.5

real = 3.2
Imaginary = -0.5

real = 4.59
Imaginary = -6.6

```

11. Write a Program to overload operators like \*, <<, >> using friend function. The following overloaded operators should work for a class vector.

CODE:

```
//Header files
#include <iostream>
#include <conio.h>
using namespace std;
//const variable size
const int size = 3;
//Vector class definition
class vector
{ //integer array v defined in default private mode
int v[size];
//access specifier
public:
//default constructor
vector();
//parameterized constructor with pointer argument
vector(int *x);
//friend function to overload operators '*' '<<' '>>'
friend vector operator*(int a, vector b);
friend vector operator*(vector b, int a);
friend ostream &operator>>(ostream &, vector &);
friend ostream &operator<<(ostream &, vector &);
};
//constructor definition to initialise values of vectors to zero
vector ::vector()
{
for (int i = 0; i < size; i++)
v[i] = 0;
}
//constructor definition to initialise values of vectors to pointer y
vector ::vector(int *y)
{
for (int i = 0; i < size; i++)
v[i] = y[i];
}
//operator overloading definition for '*' operator
vector operator*(int a, vector b)
{
vector c;
for (int i = 0; i < size; i++)
c.v[i] = a * b.v[i];
return c;
}
//operator overloading definition for '*' operator
```

```

vector operator*(vector b, int a) {
vector c;

for (int i = 0; i < size; i++)
c.v[i] = b.v[i] * a;
return c;
}
//operator overloading definition for '>>' operator
istream &operator>>(istream &din, vector &b)
{
for (int i = 0; i < size; i++)
din >> b.v[i];
return (din);
}
//operator overloading definition for '<<' operator
ostream &operator<<(ostream &dout, vector &b)
{
dout << "(" << b.v[0];
for (int i = 1; i < size; i++)
dout << "," << b.v[i];
dout << ")";
return (dout);
}
//initialising values in array x
int x[size] = {2, 4, 6};
int main()
{ //object 'm' of class vector
vector m;
vector n = x; //constructor
cout << "Enter Elements of vector m"
<< "\n";
cin >> m;
cout << "\n";
cout << "m=" << m << "\n";
//object 'p', 'q' of class vector m
vector p, q;
//called operator when first argument is int and second argument is vector
p = 2 * m;
//called operator when first argument is vector and second argument is int
q = n * 2;
cout << "\n";
cout << "p=" << p << "\n";
cout << "q=" << q << "\n";
return 0;
}

```



## OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hrds1.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Enter Elements of vector m
1
2
3

m=(1,2,3)

p=(2,4,6)
q=(4,8,12)
```

12. Write a program for developing a matrix class which can handle integer matrices of different dimensions. Also overload the operator for addition, multiplication & comparison of matrices.

CODE:

```
#include <iostream>
#include <iomanip> //It defines the manipulator functions
using namespace std;
//Matrix class definition
class matrix
{ //variable maxcol,maxrow defined in default private mode
    int maxrow, maxcol;
    //pointer variable defined in default private mode
    int *ptr;
    //access specifier
public:
    //paramaterized constructor definition
    matrix(int r, int c)
    { //value of r assigned to maxrow
        maxrow = r;
        //value of r assigned to maxcol
        maxcol = c;
        ptr = new int[r * c];
    }
    //getmat function declaration to get value in matrix
    void getmat(void);
    //printmat function declaration to print value of matrix
    void printmat();
    //delmat function declaration to find determinant value of matrix
    int delmat();
    //'+' operator overloading function declaration
    matrix operator+(matrix b);
```

```

//'*' operator overloading function declaration
matrix operator*(matrix b);

//'==' operator overloading function declaration
int operator==(matrix b);
};

//getmat function definition
void matrix ::getmat(void)
{
    int i, j, mat_off, temp;
    cout << endl
        << "enter elements matrix:" << endl;
    for (i = 0; i < maxrow; i++)
    {
        for (j = 0; j < maxcol; j++)
        { //assigning consecutive integer values to mat_off variable
            mat_off = i * maxcol + j;
            cin >> ptr[mat_off];
        }
    }
}

//printmat function declaration
void matrix ::printmat()
{
    int i, j, mat_off;
    for (i = 0; i < maxrow; i++)

    {
        cout << endl;
        for (j = 0; j < maxcol; j++)
        { //assigning consecutive integer values to mat_off variable
            mat_off = i * maxcol + j;

            //printing output in a matrix form using setw manipulator

```

```

        cout << setw(3) << ptr[mat_off];
    }
}
}
//delmat function declaration
int matrix ::delmat()
{ //constructor call
    matrix q(maxrow - 1, maxcol - 1);
    //assigning and decalaring variables
    int sign = 1, sum = 0, i, j, k, count;
    int newsize, newpos, pos, order;
    //assigning maxrow value to order
    order = maxrow;
    if (order == 1)
    { //return single value if there is only one element in matrix
        return (ptr[0]);
    }
    for (i = 0; i < order; i++, sign *= -1)
    {
        for (j = 1; j < order; j++)
        {
            for (k = 0, count = 0; k < order; k++)
            {
                if (k == i)
                    continue;
                pos = j * order + k;
                newpos = (j - 1) * (order - 1) + count;
                q.ptr[newpos] = ptr[pos];
                count++;
            }
        }
    }
    //calculating sum of elements of matrix

```

```

        sum = sum + ptr[i] * sign * q.delmat();
    }
    //returning sum of all elements
    return (sum);
}
//'+ operator overloading definition
matrix matrix ::operator+(matrix b)
{ //constructor call
    matrix c(maxrow, maxcol);

    int i, j, mat_off;
    for (i = 0; i < maxrow; i++)
    {
        for (j = 0; j < maxcol; j++)
        { //assigning consecutive integer values to mat_off variable
            mat_off = i * maxcol + j;
            //adding values of two matrix and assign it to third matrix c
            c.ptr[mat_off] = ptr[mat_off] + b.ptr[mat_off];
        }
    }
    return (c);
}
//'* operator overloading definition
matrix matrix ::operator*(matrix b)
{ //constructor call
    matrix c(b.maxcol, maxrow);
    int i, j, k, mat_off1, mat_off2, mat_off3;
    for (i = 0; i < c.maxrow; i++)
    {
        for (j = 0; j < c.maxcol; j++)
        { //assigning consecutive integer values to mat_off variable
            mat_off3 = i * c.maxcol + j;

```

```

        //assigning zero to all values of object c's pointer ptr
        c.ptr[mat_off3] = 0;
        for (k = 0; k < b.maxrow; k++)
        { //assigning values of index
            mat_off2 = k * b.maxcol + j;
            mat_off1 = i * maxcol + k;
            //multiplying matrix's element according to their index as per rule in matrix
multiplication
            c.ptr[mat_off3] += ptr[mat_off1] * b.ptr[mat_off2];
        }
    }
}
return (c);
}
//'==' operator overloading definition
int matrix ::operator==(matrix b)
{
    int i, j, mat_off;
    //checking value of row and column of a and b
    if (maxrow != b.maxrow || maxcol != b.maxcol)
        return (0);
    for (i = 0; i < maxrow; i++)
    {
        for (j = 0; j < maxcol; j++)
        { //assigning consecutive integer values to mat_off variable
            mat_off = i * maxcol + j;
            //checking condition for equality
            if (ptr[mat_off] != b.ptr[mat_off])
                return (0);
        }
    }
}
return (1);

```

```
}
```

```
int main()
{ //decalaring vriable to take value of row and column of a and b
  int rowa, cola, rowb, colb;
  cout << endl
    << "Enter dimensions of matrix A ";
  cin >> rowa >> cola;
  //constructor call with object a
  matrix a(rowa, cola);
  //getmat function call to assign inputted value using object a
  a.getmat();
  cout << endl
    << "Enter dimensions of matrix B";
  cin >> rowb >> colb;
  //constructor call with object b
  matrix b(rowb, colb);
  //getmat function call to assign inputted value using object b
  b.getmat();
  //constructor call
  matrix c(rowa, cola);
  //adding matrices using '+' overloaded operator
  c = a + b;
  cout << endl
    << "The sum of two matrices = ";
  //calling printmat function to print matrix c(constitutes addition of two matrix)
  c.printmat();
  //constructor call
  matrix d(rowa, colb);
  //multipling matrix using '*' overloaded operator
  d = a * b;
  cout << endl
```

```

<< "The product of two matrices = ";
//calling printmat function to print matrix d(constitutes multiplication of two matrix)
d.printmat());
cout << endl
    << "Determinant of matrix a =" << a.delmatt(); //calling delmatt function to cal.
determinant of matrix
//comparing matrix using '==' overloaded operator
if (a == b)
    cout << endl
        << "a & b are equal";
else
    cout << endl
        << "a & b are not equal";
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

Enter dimensions of matrix A 2
2

enter elements matrix:
1
2
3
4

Enter dimensions of matrix B2
2

enter elements matrix:
1
2
3
4

The sum of two matrices =
 2  4
 6  8
The product of two matrices =
 7 10
15 22
Determinant of matrix a =-2
a & b are equal

```



13. Write a program to overload new / delete operators in a class.

CODE:

```
//header file for standard input and output
#include <iostream>

//header file mainly for dynamic memory and general functions
#include <stdlib.h>

using namespace std;

//student class definition
class student
{
    //variables defined in private
    string name;
    int age;
    //access specifier public
public:
    //default constructor
    student()
    {
        cout << "Constructor is called\n";
    }
    //parameterized constructor with parameter name and age
    student(string name, int age)
    {
        //assigning name and age variable using this pointer as the name is same
        this->name = name;
        this->age = age;
    }
    //display function definition
```

```

void display()
{

    cout << "Name:" << name << endl;
    cout << "Age:" << age << endl;
}

//overloading new operator
void *operator new(size_t size)
{
    cout << "Overloading new operator with size: " << size << endl;
    void *p = ::new student();
    //void * p = malloc(size); will also work fine
    return p;
}

//overloading delete operator
void operator delete(void *p)
{
    cout << "Overloading delete operator " << endl;
    free(p);
}
};

int main()
{
    //using new operator
    student *p = new student("Yash", 24);
    //printing output
    p->display();
    //deleting output
    delete p;
}

```

```
return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
Overloading new operator with size: 28  
Constructor is called  
Name:Yash  
Age:24  
Overloading delete operator
```

14. Write a program in C++ to highlight the difference between the overloaded assignment operator and copy constructor.

CODE:

```
#include <iostream>
using namespace std;
//CIRCLE class definition
class circle
{
    //private access specifier
private:
    //declaring variables
    int radius;
    float x, y;
    //public access specifier
public:
    //default constructor declaration
    circle()
    {
    }
    //paramaterized constructor declaration
    circle(int rr, float xx, float yy)
    { //assigning value of rr to radius
        radius = rr;
        //assigning value of xx to x
        x = xx;
        //assigning value of yy to y
        y = yy;
        cout << endl
            << "simple constuctor called";
    }
    //'=' Assignment operator overloading definition
```

```

circle operator=(const circle &c)
{
    cout << endl
        << "Assignment operator invoked";
    radius = c.radius;
    x = c.x;
    y = c.y;
    return circle(radius, x, y);
}
//copy constructor definition
circle(const circle &c)
{
    cout << endl
        << "copy constructor invoked";
    radius = c.radius;
    x = c.x;

    y = c.y;
}
//showdata function definition to print output
void showdata()
{
    cout << endl
        << "Radius = " << radius;
    cout << endl
        << "X-Coordinate=" << x;
    cout << endl
        << "Y-Coordinate=" << y;
}
};
int main()
{

```

```

circle c1(10, 2.5, 2.5); //parametrized constructor called
circle c2, c4;      // null constructor
c4 = c2 = c1;      //assignment,function,assignment,function
circle c3 = c1;    //copy constructor
//calling showdat function to print output
c1.showdata();
c2.showdata();
c3.showdata();
c4.showdata();
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

```

```

simple constuctor called
Assignment operator invoked
simple constuctor called
Assignment operator invoked
simple constuctor called
copy constructor invoked
Radius = 10
X-Coordinate=2.5
Y-Coordinate=2.5
Radius = 10
X-Coordinate=2.5
Y-Coordinate=2.5
Radius = 10
X-Coordinate=2.5
Y-Coordinate=2.5
Radius = 10
X-Coordinate=2.5
Y-Coordinate=2.5

```

15. Write a Program illustrating how the constructors are implemented and the order in which they are called when the classes are inherited. Use three classes named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha & beta.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;
//base alpha class definition
class alpha
{ //variable x defined in default private mode
    int x;
    //public access specifier
public:
    //parameterized constructor
    alpha(int i)
    {
        x = i;
        cout << "alpha initialized\n";
    }
    //show_x function definition
    void show_x(void)
    {
        cout << "x=" << x << "\n";
    }
};
//base beta class definition
class beta
{
    //variable y defined in private default mode
    float y;
    //public access specifier
public:
    //parameterized constructor
    beta(float j)
    {
        y = j;
        cout << "beta initialized\n";
    }
    // //show_y function definition
```

```

void show_y(void)
{
    cout << "y= " << y << "\n";
}
};
//derived gamma class definition
class gamma : public beta, public alpha
{
    int m, n;
    //public access specifier
public:
    //parameterized constructor
    gamma(int a, float b, int c, int d) : alpha(a), beta(b)
    {
        m = c;
        n = d;
        cout << "gamma initialized\n";
    }
    //show_mn function definition
    void show_mn(void)
    {

        cout << "m=" << m << "\n";
        cout << "n=" << n << "\n";
    }
};
int main()
{ //constructor called
    gamma g(5, 10.75, 20, 30);
    //show_x function call
    g.show_x();
    //show_y function call
    g.show_y();
    //show_mn function call
    g.show_mn();
    return 0;
}

```



OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
beta initialized
alpha initialized
gamma initialized
x=5
y= 10.75
m=20
n=30
```

16. Write a Program to design a student class representing student roll no. and a test class (derived class of student) representing the scores of the student in various subjects and sports class representing the score in sports. The sports and test class should be inherited by a result class having the functionality to add the scores and display the final result for a student.

CODE:

```
//Header file for standard input and output function
#include <iostream>
using namespace std;
//Base student class
class student
{
    //protected access specifier
protected:
    //int variable roll_number
    int roll_number;
    //public access specifier
public:
    //get_number function definition to take value of roll_no. from user
    void get_number(int a)
    { //value of 'a' assigned to roll_number
        roll_number = a;
    }
    //put_number function definition to print output of roll no.
    void put_number(void)
    {
        cout << "Roll No:" << roll_number << "\n";
    }
};
//Derived test Class publicly from student class
class test : public student
```

```

{
    //protected access specifier
protected:
    float part1, part2;
    //public access specifier
public:
    //get_marks function definition to take two subject's marks from user
    void get_marks(float x, float y)
    { //assigning value of marks in part1 and part2
        part1 = x;
        part2 = y;
    }
    //put_marks function definition to print values/marks of two subjects
    void put_marks(void)
    {
        cout << "Marks obtained"
            << "\n"
            << "part1 =" << part1 << "\n"
            << "part2 =" << part2 << "\n";
    }
};
//Base sports class
class sports
{
    //protected access specifier
protected:
    float score;
    //public access specifier
public:

```

```

//get_score function definition to take sports subject marks from user
void get_score(float s)
{ //assigning sports marks 's' to score
    score = s;
}

//put_score function definition to print the marks of sport subject
void put_score(void)
{
    cout << "Sports wt:" << score << "\n\n";
}
};

//Derived result class publicly from test and sports class
class result : public test, public sports
{
    //total variable defined in default private mode
    float total;
    //public access specifier
public:
    //display function declaration
    void display(void);
};

//display function definition
void result ::display(void)
{ //suming up marks of all subject and assigned it to total variable
    //we are able to use part1,part2 and score variable as derived class can access all
    the non - private members of its base class
        total = part1 + part2 + score;
    //calling put_number function of 'test's' base class 'student'
    put_number();
}

```

```

//calling put_marks function of 'result's' base class 'test'
put_marks();
//calling put_score function of 'sports' base class
put_score();
//print total of all subjects
cout << "Total Score:" << total << "\n";
}
int main()
{ //student_1 object declaration of class result
result student_1;
//functions to assign value
//calling get_number funtion
student_1.get_number(1234);
//calling get_marks funtion
student_1.get_marks(27.5, 33.0);
//calling get_score funtion
student_1.get_score(6.0);
//function to print values
//calling display funtion to diplay all the output
student_1.display();
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hrds1.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Roll No:1234
Marks obtained
part1 =27.5
part2 =33
Sports wt:6

Total Score:66.5

```

17. Write a program to maintain the records of person with details (Name and Age) and find the eldest among them. The program must use this pointer to return the result. Header file for standard input and output file

CODE:

```
#include <iostream>
//Header file to include string
#include <string.h>
using namespace std;
//person class definiton
class person
{
    char name[20];
    float age;
    //public access specifier
public:
    person(const char *s, float a) //const before char*-in C the type is array of char and in C++
it is constant array of char
    {
        strcpy(name, s);
        age = a;
    }
    //greater function definition
    person &greater(person &x)
    {
        if (x.age >= age)
            return x;
        else
            return *this;
    }
    //display function definition
    void display(void)
    {
        cout << "Name:" << name << "\n"
        << "Age: " << age << "\n";
    }
};
int main()
{ //constructor call
    person p1("John", 37.50), p2("Ahmed", 29.0), p3("Hebber", 40.5);
    person p = p1.greater(p3);
```

```
cout << "Elder Person is:\n";  
//display function call  
p.display();  
//greater function call  
p = p1.greater(p2);  
  
cout << "Elder Person is:\n";  
//display function call  
p.display();  
return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
Elder Person is:  
Name:Hebber  
Age: 40.5  
Elder Person is:  
Name:John  
Age: 37.5
```

18. Write a Program to illustrate the use of pointers to objects which are related by inheritance.

CODE:

```
//Header file for standard input output
#include <iostream>
using namespace std;
//Base class BC definition
class BC
{
    //access specifier
public:
    //variable b declaration in public mode
    int b;
    //show function definition
    void show()
    {
        cout << "b=" << b << "\n";
    }
};
//Derived class DC definition
class DC : public BC
{
    //access specifier
public:
    //variable d declaration in public mode
    int d;
    //show function definition
    void show()
    {
        cout << "b=" << b << "\n"
            << "d=" << d << "\n";
    }
};
int main()
{
    BC *bptr; //base class pointer
    BC base; //base class object
    //case 1:Using base class pointer to assign base class object
    bptr = &base; // base object's address assigned to base pointer
```



```

bptr->b = 100; //base pointer access base variable in base object
cout << "bptr points to base object\n";
bptr->show(); //show of base class

//case 2:Using base class pointer to assign derived class object
DC derived; // derived class object
bptr = &derived; // derived object's address assigned to base pointer
bptr->b = 200; //base pointer access base variable in derived object
cout << "bptr now points to derived object\n";
bptr->show(); // base pointer access base member function in derived object

//case 3: Using derived class pointer to assign derived class object
DC *dptr; //derived class pointer
dptr = &derived; // derived object's address assigned to derived pointer
dptr->d = 300; //derived pointer access derived object personal member variable
cout << "dptr is derived type pointer\n";
dptr->show(); //derived pointer access derived personal member function

//case4: //Using base class pointer to assign derived class object with the help of
typecasting
cout << "Using ((DC *)bptr)\n";
((DC *)bptr)->d = 400; //using base pointer to call derived member variable
((DC *)bptr)->show(); // using base pointer to call derived member function
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
bptr points to base object
b=100
bptr now points to derived object
b=200
dptr is derived type pointer
b=200
d=300
Using ((DC *)bptr)
b=200
d=400

```

19. Write a program illustrating the use of virtual functions in class.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;
//Base class definition
class Base
{
    //access specifier
public:
    //Display function definition
    void display()
    {
        cout << "\n Display Base";
    }
    //show function definition derived as virtual
    virtual void show() // base class member function derived as virtual
    {
        cout << "\n Show Base:";
    }
};
//Derived class definition derived publicly from base class
class Derived : public Base
{
    //access specifier
public:
    //display function definition
    void display()
    {
        cout << "\n Display Derived";
    }
    //show function definition
    void show()
    {
        cout << "\n Show Derived";
    }
};
int main()
{ //base class object declaration
```

```

Base B;
//derived class object declaration
Derived D;
//Base class pointer declaration
Base *bptr;
cout << "\n bptr points to Base\n";
// base class object's address assigned to base pointer
bptr = &B;
//display function calling of base class
bptr->display();
//show function calling of base class
bptr->show();
cout << "\n\n bptr points to derived\n";
// derived class object's address assigned to base pointer
bptr = &D;
bptr->display(); // base class member function is called by base class pointer
bptr->show(); //derived class member function is called by base class pointer
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

bptr points to Base

Display Base
Show Base:

bptr points to derived

Display Base
Show Derived

```

20. Write a program to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media ). The class should have the functionality for adding new item, issuing, deposit etc. The program should use the runtime polymorphism.

CODE:

```
#include <bits/stdc++.h>
using namespace std;

class media
{
protected:
    char title[50];
    float price;

public:
    media(char *s, float a)
    {
        strcpy(title, s);
        price = a;
    }
    virtual void display() {}
};

class book : public media
{
    int pages;

public:
    book(char *s, float a, int p) : media(s, a)
    {
        pages = p;
    }
    void display();
};

class tape : public media
{
    float time;

public:
    tape(char *s, float a, float t) : media(s, a)
```

```

    {
        time = t;
    }
    void display();
};
void book::display()
{
    cout << "Title:" << title << "\n";
    cout << "Pages:" << pages << "\n";
    cout << "Price:" << price << "\n"
        << "\n";
}
void tape::display()
{
    cout << "Title:" << title << "\n";
    cout << "Play Time:" << time << " "
        << "mins"
        << "\n";
    cout << "Price:" << price << "\n"
        << "\n";
}
int main()
{

    char *title = new char[30];
    float price, time;
    int pages;

    cout << "Enter Book Details"
        << "\n";
    cout << "Title:";
    cin >> title;
    cout << "Price:";
    cin >> price;
    cout << "Pages:";
    cin >> pages;

    book book1(title, price, pages);

    cout << "Enter Tape Details"
        << "\n";
}

```

```

cout << "Title: ";
cin >> title;
cout << "Price: ";
cin >> price;
cout << "Play Time in mins: ";
cin >> time;

tape tape1(title, price, time);

media *list[2];
list[0] = &book1;
list[1] = &tape1;
cout << "\n";
cout << "Media Details"
    << "\n"
    << "\n";
cout << "<-Book->"
    << "\n"
    << "\n";
list[0]->display();
cout << "<-Tape->"
    << "\n"
    << "\n";
list[1]->display();

return 0;
}

```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Enter Book Details
Title:Harry_Potter
Price:400
Pages:1000
Enter Tape Details
Title:Hey_Ram
Price:40
Play Time in mins:10

Media Details

<-Book->

Title:Harry_Potter
Pages:1000
Price:400

<-Tape->

Title:Hey_Ram
Play Time:10 mins
Price:40
```

21. Write a program to show conversion from string to int and vice-versa.

CODE:

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;
//String Class definition
class String
{
    //private access specifier
private:
    char str[20];
    //public access specifier
public:
    //Default Constructor definition
    String()
    {
        str[0] = '\0';
    }
    //Parameterized Constructor's definition
    //copy character array into string
    String(const char *s)
    {
        strcpy(str, s);
    }
    //Parameterized Constructor's definition
    //converts integer to string
    String(int a)
    {
        itoa(a, str, 10);
    }
    //converts string into integer
    operator int() //conversion operator, Conversion operator doesn't have any return type
    not even void.
    {
        int i = 0, l, ss = 0, k = 1;
        l = strlen(str) - 1;
        while (l >= 0)
        {
            ss = ss + (str[l] - 48) * k;
```



```

        l--;
        k *= 10;
    }
    return (ss);
}
//display data function definition
//prints the output string
void displaydata()
{
    cout << str;
}
};
int main()
{
    //constructor call
    String s1 = 123; //iota function, conversion from int to "String"

    cout << endl
        << "s1=";
    //displaydata function call
    s1.displaydata();
    //constructor call
    s1 = 150; //iota function, operand type are "String" and int
    cout << endl
        << "s1=";
    //displaydata function call
    s1.displaydata();
    String s2("123"); // parametrize constructor
    int i = int(s2); //converting "String" to int, explicit call to String::operator int()
    cout << endl
        << "i=" << i;
    String s3("456"); // parametrize constructor
    i = s3; //converting "String" to int, this will implicitly call String::operator int()
    cout << endl
        << "i=" << i;
    return 0;
}

```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

s1=123
s1=150
i=123
i=456
```

22. Write a program showing data conversion between objects of different classes.

CODE:

```
//Header file for standard input and output
#include <iostream>
//Header file for "itoa" function
#include <stdlib.h>
//Header file for strings functions
#include <string.h>
using namespace std;
//date class definition
//to store str into dt array
class date
{
    //private access specifier
private:
    //dt array to store date in "/" format
    char dt[9];
    //public access specifier
public:
    //default constructor
    date()
    {
        dt[0] = '\0';
    }
    //parameterized constructor
    date(char *s)
    {
        strcpy(dt, s);
    }
    //displaydata function to print output
    void displaydata()
    {
        cout << dt;
    }
};
//dmy class definition
//to convert separate date,month,year into "/" format
class dmy
{
    //private access specifier
```

```

private:
    //variables to store date,month,year respectively
    int day, mth, yr;
    //public access specifier
public:
    //default constructor
    dmy()
    {
        day = mth = yr = 0;
    }
    //parameterized constructor
    dmy(int d, int m, int y)

    {
        day = d;
        mth = m;
        yr = y;
    };
    //casting constructor operator function to change data type
    operator date()
    {
        char temp[3], str[9];
        itoa(day, str, 10);
        //concatinating "/" in char array
        strcat(str, "/");
        itoa(mth, temp, 10);
        //concatinating temp value in char array
        strcat(str, temp);
        //concatinating "/" in char array
        strcat(str, "/");
        itoa(yr, temp, 10);
        //concatinating temp value in char array
        strcat(str, temp);
        return (date(str));
    }
    //displaydata function to print output
    void displaydata()
    {
        cout << day << "\t" << mth << "\t" << yr;
    }
};

```

```

int main()
{ //Object creation of date
  // Default constructor called automatically
  date d1;
  // Constructor called of dmy
  dmy d2(17, 11, 94);
  //Assigning value of d2 into d1
  d1 = d2; // or we can write d1=date(d2), but we have not written dmy(reverse) conversion
function
  cout << endl
    << "d1=";
  //displaydata function call for date
  d1.displaydata();
  cout << endl
    << "d2=";
  //displaydata function call for dmy
  d2.displaydata();
  return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

d1=17/11/94
d2=17  11    94

```

23. Write a program showing data conversion between objects of different classes and conversion routine should reside in destination class.

CODE:

```
//Header file for standard input and output
#include <iostream>
//Header file for string functions
#include <string.h>
//Header file for itoa function
#include <stdlib.h>
using namespace std;
//dmy class function(source class)
class dmy
{
    //variable to store date,month and year respectively in default private mode
    int day, mth, yr;

public:
    //default constructor
    dmy()
    {
        day = mth, yr = 0;
    }
    //parameterized constructor
    dmy(int d, int m, int y)
    {
        day = d;
        mth = m;
        yr = y;
    }
    //getday function definition to return day
    int getday()
    {
        return (day);
    }
    //getmth function definition to return month
    int getmth()
    {
        return (mth);
    }
    //getyr function definition to return year
```

```

int getyr()
{
    return (yr);
}
//displaydata function definition to print output
void displaydata()
{
    cout << day << "\t" << mth << "\t" << yr;
}
};
//date class function(destination class)
class date
{
    //private access specifier
private:
    char dt[9];
    //public access specifier
public:
    //default constructor
    date()
    {
        dt[0] = '\0';
    }
    //parameterized constructor
    date(char *s)
    {
        strcpy(dt, s);
    }
    //displaydata function definition to print output
    void displaydata()
    {
        cout << dt;
    }
    //casting operator function
    date(dmy t)
    { //assigning value of date,month and year in respective variables
        int d = t.getday();
        int m = t.getmth();
        int y = t.getyr();
        char temp[3];
        //calling itoa function

```

```

        itoa(d, dt, 10);
        //concatinating "/" in char array
        strcat(dt, "/");
        //calling itoa function
        itoa(m, temp, 10);
        //concatinating temp value in char array
        strcat(dt, temp);
        //concatinating "/" in char array
        strcat(dt, "/");
        //calling itoa function
        itoa(y, temp, 10);
        //concatinating temp value in char array
        strcat(dt, temp);
    }
};
int main()
{
    //Object creation of date
    // Default constructor called automatically
    date d1;
    // Constructor called of dmy
    dmy d2(17, 11, 94);
    //Assigning value of d2 into d1
    d1 = d2;
    cout << endl
        << "d1=";
    //displaydata function call for date
    d1.displaydata();
    cout << endl
        << "d2=";
    //displaydata function call for dmy
    d2.displaydata();

    return 0;
}

```



OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
```

```
d1=17/11/94
d2=17  11    94
```

24. Write a program implementing basic operation of class ios i.e. setf, unsetf, precision etc.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;
int main()
{ //varibale declaration
  int i = 52;
  float a = 425.0;
  float b = 123.500328;
  char str[] = "Dream. Then make it happend!";

  cout.setf(ios::unitbuf); //When the unitbuf flag is set, the associated buffer is flushed after
  each insertion operation.
  //cout.setf( ios::stdio );
  cout.setf(ios::showpos); //When the showpos format flag is set, a plus sign (+)
  precedes every non - negative numerical value inserted into the stream(including zeros)
  cout << i << endl; //output-+52
  cout.setf(ios::showbase); //When the showbase format flag is set, numerical integer
  values inserted into output streams are prefixed with the same prefixes used by C++ literal
  constants :
  cout.setf(ios::uppercase); //uppercase (capital) letters are used instead of lowercase
  cout.setf(ios::hex, ios::basefield); //0x for hexadecimal values
  cout << i << endl; //output-0X34
  cout.setf(ios::oct, ios::basefield); //0 for octal values and no prefix for decimal-base values
  cout << i << endl; //output-064
  cout.setf(ios::dec, ios::basefield); // no prefix for decimal-base values
  cout << i << endl; //output-+52

  cout.fill('0'); // sets 0 as the new fill character and returns the fill
  character used before the call. The fill character is the character used by output insertion
  functions to fill spaces when padding results to the field width
  cout << "Fill character " << cout.fill() << endl; // output - Fill Character 0 //cout.fill( ) here
  returns the fill character.

  cout.width(10); //sets a new field width for the stream
  cout << i << endl; //output-0000000+52
  cout.setf(ios::left, ios::adjustfield); //Sets the adjustfield format flag for the str stream to
  left.The adjustfield format flag can take any of the following values-internal, left, right
  cout.width(10);
```

```

cout << i << endl;           //+520000000
cout.setf(ios::internal, ios::adjustfield); // magnitude to right, sign to left
cout.width(10);              // set width again otherwise print in default width
cout << i << endl;           ////+000000052
cout << endl;                 //newline
cout << endl;                 //newline
cout.width(10);
cout << str << endl; //Dream. Then make it happend!
cout.width(40);
cout.setf(ios::left, ios::adjustfield);
cout << str << endl;         // Dream. Then make it happend!000000000000
cout.precision(6);           // sets it to a new value
cout << "Precision" << cout.precision(); //output-Precision+6 //cout.precision( ) returns
the value of the current floating-point precision field for the stream
cout.setf(ios::showpoint);    // the decimal point is always written for floating point
values inserted into the stream (even for those whose decimal part is zero)
cout.unsetf(ios::showpos);     //remove the +sign from non egative number
cout << endl
    << a; //425.000
cout.unsetf(ios::showpoint);
cout << endl
    << a;           //425
cout.setf(ios::fixed, ios::floatfield); // floatfield is set to fixed, floating-point values are
written using fixed-point notation
cout << endl
    << b;           //123.500328
cout.setf(ios::scientific, ios::floatfield); //write floating-point values in scientific notation.
cout << endl
    << b; //1.235003E+002
b = 5.375;
cout.precision(14);
cout.setf(ios::fixed, ios::floatfield);

cout << endl
    << b; //5.3750000000000000
cout.setf(ios::scientific, ios::floatfield);
cout << endl
    << b; //5.3750000000000000E+000
cout.unsetf(ios::showpoint);
cout.unsetf(ios::unitbuf);
// cout.unsetf( ios::stdio );

```

```
    return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
+52  
0X34  
064  
+52  
Fill character 0  
000000+52  
+52000000  
+00000052  
  
Dream. Then make it happend!  
Dream. Then make it happend!000000000000  
Precision+6  
425.000  
425  
123.500328  
1.235003E+002  
5.37500000000000  
5.37500000000000E+000
```

25. Write a program to implement I/O operations on characters. I/O operations includes input a string, length of string, store it in a file, fetching the stored characters from it, etc.

CODE:

```
//Header file to include standard input and output function
#include <iostream>
//Header file to include function to do file handling
#include <fstream>
//Header file to include string functions
#include <string.h>
using namespace std;
int main()
{ //variable declaration
  char String[80];
  char ch;
  cout << "Enter a String \n";
  cin >> String;
  //storing size of inputted string in variable "len"
  int len = strlen(String);
  //input and output file object declaration
  fstream file;
  //opening file
  file.open("check.txt", ios::in | ios::out); // u can give whole path to file name but add '/'
  with every other '/'

  for (int i = 0; i <= len; i++)
  {
    //writing/putting value in file
    file.put(String[i]);
  }

  //change cursor position to 0th/starting position using seekg
  file.seekg(0);
  while (!file.eof())
  { //retrieving data from file using get
    file.get(ch);
    cout << ch;
  }
  //closing file
  file.close();
}
```

```
    return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
Enter a String  
Solitude  
Solitude
```



26. Write a program to copy the contents of one file to another.

CODE:

```
//Header file to include function to create and write on file
#include <fstream>
//Header file to include standard input and output functions
#include <iostream>
using namespace std;
int main()
{
    //Declaring char variable to store file name and use it
    char source[67], target[67];
    //declaring ch variable to write and get chracters in file
    char ch;
    cout << endl
        << "Enter source filename" << endl;
    cin >> source;
    cout << endl
        << "Enter target filename" << endl;
    cin >> target;
    //object declaration infile to take input from source file
    ifstream infile(source);
    //object declaration outfile to write something or put in output file
    ofstream outfile(target);
    while (infile)
    { //getting from source file
        infile.get(ch);
        //putting in output file
        outfile.put(ch);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
```

```
Enter source filename
check.txt
```

```
Enter target filename
answer.txt
```





27. Write a program to perform read/write binary I/O operation on a file (i.e. write the object of a structure/class to file).

CODE:

```
//Header file to include function to create and write on file
#include <fstream>
//Header file to include standard input and output functions
#include <iostream>
using namespace std;
int main()
{ //declaring structure named employee
  struct employee
  { //declaring variables
    char name[20];
    int age;
    float basic;
    float gross;
  };
  //declaring variable e using employee data type
  employee e;
  //storing "Y" in ch for conditioning
  char ch = 'Y';
  //decalaring object "outfile" to put data in file
  ofstream outfile;
  //opening file
  outfile.open("EMPLOYEE.txt", ios::out | ios::binary);
  while (ch == 'Y')
  {
    cout << endl
      << "Enter a record - name, age, basic salary, gross salary -";
    cin >> e.name >> e.age >> e.basic >> e.gross;
    //writing into file
    outfile.write((char *)&e, sizeof(e));
    cout << endl
      << "Add Another Y/N" << endl;
    cin >> ch;
  }
  //closing file
  outfile.close();
  //decalaring object "outfile" to take data from file
```

```

ifstream infile;
//opening file
infile.open("EMPLOYEE.txt", ios::in | ios::binary);
cout << endl
    << "Name"
    << "\t"
    << "Age"
    << "\t"
    << "Basic"
    << "\t"
    << "Gross";
//reading data from file
while (infile.read((char *)&e, sizeof(e)))
{
    cout << endl
        << e.name << "\t" << e.age << "\t" << e.basic << "\t" << e.gross;
}
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

Enter a record - name, age, basic salary, gross salary -Shirley
26
100000
10000000

Add Another Y/N
n

Name    Age    Basic    Gross
Shirley 26     100000  1e+007

```

28. Write a program to maintain a elementary database of employees using files.

CODE:

```
//Header file to include specified functions in program
#include <windows.h>
#include <fstream>
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <iomanip>
using namespace std;
//Class group definition
class group
{
    //private access specifier
private:
    //Person struct definition
    struct person
    {
        char flag;
        char empcode[5];
        char name[40];
        int age;
        float sal;
    } p; //object creation
    //file stream "file" object declaration
    fstream file;
    //public access specifier
public:
    //constructor declaration
    group();
    //all functions declaration
    void addrec();
    void listrec();
    void modirec();
    void delrec();
    void exit();
};
```

```

int main()
{

    char choice;
    group g;

    do
    {
        //clrscr();
        system("cls");

        cout << "1. Add records" << endl;
        cout << "2. List records" << endl;
        cout << "3. Modify records" << endl;
        cout << "4. Delete records" << endl;
        cout << "0. Exit" << endl;
        cout << "Your Choice ? " << endl;
        cin >> choice;
        system("cls");
        //Switching between cases to create menu options
        switch (choice)
        {
            case '1':
                g.addrec();
                break;
            case '2':
                g.listrec();
                break;
            case '3':
                g.modirec();
                break;
            case '4':
                g.delrec();
                break;
            case '0':
                g.exit();
                break;
        }
    } while (choice != '0');
    return 0;
}

```

```

//constructor definition
group::group()

{
    //opening file
    file.open("file.txt", ios::out | ios::binary | ios::in); // first out then binary then in, other
wise file wont open
    //checking availability of file
    if (!file)
    {
        cout << endl
            << "Unable to open file";
        exit();
    }
}
//add record function definition
void group::addrec()
{
    char ch;
    //moving pointer to end to add new record at the end
    file.seekp(0L, ios::end);
    do
    {
        cout << endl
            << "Enter emp code, name, age & salary" << endl;
        cin >> p.empcode >> p.name >> p.age >> p.sal;
        p.flag = ' ';
        //inputting/writing data in the file
        file.write((char *)&p, sizeof(p));
        cout << "Add another record? (Y/N)";
        cin >> ch;
    } while (ch == 'Y' || ch == 'y');
}
//list record function definition
void group::listrec()
{
    int j = 0, a;
    //brings the pointer to beginning
    file.seekg(0L, ios::beg);
    //reading data to its size
    while (file.read((char *)&p, sizeof(p)))
    {

```

```

        if (p.flag != '*')
        {
            cout << endl
                << "Record#" << j++ << setw(6) << p.empcode << setw(20) << p.name << setw(4)
<< p.age << setw(9) << p.sal;
        }
        file.clear();
        cout << endl
            << "Press any key.....";

        getch();
    }
}
//modify record function definition
void group::modirec()
{
    char code[5];
    int count = 0;
    long int pos;
    cout << "Enter employee code: ";
    cin >> code;
    file.seekg(0L, ios::beg);
    while (file.read((char *)&p, sizeof(p)))
    { //if entered code is equal to present data then modify
        if (strcmp(p.empcode, code) == 0)
        {
            cout << endl
                << "Enter new record" << endl;
            cin >> p.empcode >> p.name >> p.age;
            p.flag = ' ';
            pos = count * sizeof(p);
            file.seekp(pos, ios::beg);
            file.write((char *)&p, sizeof(p));
            return;
        }
        count++;
    }
    cout << endl
        << "No employee in file with code = " << code;
    cout << endl
        << "Press any key .....";
}

```

```

    getch();
    file.clear();
}
void group::delrec()
{
    char code[5];
    long int pos;
    int count = 0;
    cout << "Enter employee code : ";
    cin >> code;
    file.seekg(0L, ios::beg);

    while (file.read((char *)&p, sizeof(p)))
    {
        if (strcmp(p.empcode, code) == 0)
        {
            p.flag = '*';
            pos = count * sizeof(p);
            //moving pointer at the beginning
            file.seekp(pos, ios::beg);
            //writing in the file
            file.write((char *)&p, sizeof(p));
            return;
        }
        count++;
    }
    cout << endl
        << "No employee in file with code = " << code;
    cout << endl
        << "Press any key ....";
    getch();
    file.clear();
}
//close function definition
void group::exit()
{
    file.close();
}

```

OUTPUT: (Removing system("cls")) to just print the output on one screen)

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
1. Add records
2. List records
3. Modify records
4. Delete records
0. Exit
Your Choice ?
2

Record#0      0          harsh  19   5e+006
Press any key.....1. Add records
2. List records
3. Modify records
4. Delete records
0. Exit
Your Choice ?
1

Enter emp code, name, age & salary
1
Yash
16
1000000
Add another record? (Y/N)N
1. Add records
2. List records
3. Modify records
4. Delete records
0. Exit
Your Choice ?
0
```



29. Write a Program for reading and writing data to and from the file using command line arguments.

CODE:

```
//Header file to include standard input and output file/functions
#include <iostream>
//Header file to include files functions
#include <fstream>
#include <stdlib.h>
using namespace std;
//Main function with command line arguments
int main(int argc, char *argv[])
{ //Pre=defined array
    int number[9] = {11, 22, 33, 44, 55, 66, 77, 88, 99};
    int i;
    //condition to check less arguments input
    if (argc != 3)
    {
        cout << "argc=" << argc << "\n";
        cout << "Error in arguments\n";
        exit(1);
    }
    //output stream object declaration
    ofstream fout1, fout2;
    //opening argument1 file
    fout1.open(argv[1]);
    //checking error in file opening
    if (fout1.fail())
    {
        cout << "Could not open the file:"
            << argv[1] << "\n";
        exit(1);
    }
    //opening argument2 file
    fout2.open(argv[2]);
    //checking error in file opening
    if (fout2.fail())
    {
        cout << "Could not open the file:"
            << argv[2] << "\n";
    }
}
```

```

        exit(1);
    }
    for (i = 0; i < 9; i++)
    { //If the number is even put it in argument1 file
        if (number[i] % 2 == 0)
            fout2 << number[i] << " ";
        else //else in argument2 file

            fout1 << number[i] << " ";
    }
    //closing files
    fout1.close();
    fout2.close();
    //input stream object declaration
    ifstream fin;
    char ch;
    //Printing the output
    for (i = 1; i < argc; i++)
    {
        fin.open(argv[i]);
        cout << "Contents of " << argv[i] << "\n";
        do
        {
            fin.get(ch);
            cout << ch;
        } while (fin);
        cout << "\n\n";
        fin.close();
    }
    return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o 3
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./3 emap.txt text.txt
Contents of emap.txt
11 33 55 77 99

Contents of text.txt
22 44 66 88

```

30. Write a program showing implementation of stack class having the functionality of push, pop operations.

CODE:

```
//Header file to include standard input output function
#include <iostream>
//defining value 10 to Max
#define MAX 10
using namespace std;
//Stack class definition
class stack
{
    //private access specifier
private:
    int arr[MAX], top;
    //public access specifier
public:
    //default constructor definition
    stack()
    {
        top = -1;
    }
    //push function definition to push elements in stack
    void push(int item)
    { //condition to check whether the stack is full or not
        if (top == MAX - 1)
        {
            cout << endl
                << "Stack is full";
            return;
        }
        //counting the number of element get pushed in stack
        top++;
        //pushing element on top
        arr[top] = item;
    }
    //pop function to delete elements in the stack
    int pop()
    { //condition to check whether the stack is empty or not
        if (top == -1)
```

```

    {
        cout << endl
            << "Stack is empty";
        return 0;
    }
    //storing top element in data
    int data = arr[top];
    //decrementing stack index value
    top--;
    return data;
}
};
int main()
{
    stack s;
    //pushing element in stack
    //call push function
    s.push(11);
    s.push(12);
    s.push(13);

    s.push(14);
    s.push(15);
    s.push(16);
    s.push(17);
    s.push(18);
    s.push(19);
    s.push(20);
    //this 21 item will not get pushed as the limit 10 of stack has reached
    s.push(21);
    //storing top value of stack in i variable
    int i = s.pop();
    //printing popped item
    cout << endl
        << "Item popped=" << i;
    i = s.pop();
    cout << endl
        << "Item popped=" << i;
    i = s.pop();
    cout << endl
        << "Item popped=" << i;
}

```

```
i = s.pop();  
cout << endl  
    << "Item popped=" << i;  
return 0;  
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap  
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap  
  
Stack is full  
Item popped=20  
Item popped=19  
Item popped=18  
Item popped=17
```

31. Write program to implement a queue class with required operations functions.

CODE:

```
//Header file to include standard input and output
```

```
#include <iostream>
```

```
//Defining 10 to MAX
```

```
#define MAX 10
```

```
using namespace std;
```

```
//queue class definition
```

```
class queue
```

```
{
```

```
    //private access specifier
```

```
private:
```

```
    //decalaring array and front and rear index
```

```
    int arr[MAX];
```

```
    int front,
```

```
        rear;
```

```
    //public access specifier
```

```
public:
```

```
    //default constructor
```

```
    queue()
```

```
{
```

```
    front = -1;
```

```
    rear = -1;
```

```
}
```

```
    //addq function definition
```

```
    void addq(int item)
```

```
{
```

```

//condition to check queue is full or not
if (rear == MAX - 1)
{
    cout << endl
        << "Queue is full";
    return;
}
rear++;
//adding item in array or queue
arr[rear] = item;
//fixing front index at 0
if (front == -1)
    front = 0;
}
//delq function definition
int delq()
{
    int data;
    //condition to check queue is empty or not
    if (front == -1)
    {
        cout << endl
            << "Queue is empty";
        return 0;
    }
    //storing deleting element in data variable
    data = arr[front];
    //to check element is last or not

```

```
    if (front == rear)
        front = rear = -1;
    else
        front++;
    return data;
}
};
```

```
int main()
{ //object creation
    queue a;
    //adding element in queue
    a.addq(11);
    a.addq(12);
    a.addq(13);
    a.addq(14);
    a.addq(15);
    a.addq(16);
    a.addq(17);
    a.addq(18);
    a.addq(19);
    a.addq(20);
    //cannot be added because queue is full
    a.addq(21);
    //deleting element
    int i = a.delq();
    cout << endl
        << "Item deleted=" << i;
```



```
//deleting element
i = a.delq();
cout << endl
    << "Item deleted=" << i;
//deleting element
i = a.delq();
cout << endl
    << "Item deleted=" << i;
return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Queue is full
Item deleted=11
Item deleted=12
Item deleted=13
```

32. Write a program to implement circular queue class with required operations/ functions.

CODE:

```
//Header file to include standard input and output file
#include <iostream>
//defining MAX to be 10
#define MAX 10
using namespace std;
//queue class definition
class queue
{
    //private access specifier
private:
    int arr[MAX];
    int front, rear;
    //public access specifier
public:
    //default constructor
    queue()
    {
        front = -1;
        rear = -1;
    }
    //addq function definition
    void addq(int item)
    { //condition to check whether the queue is full or not
        if ((rear == MAX - 1 && front == 0) || (rear + 1 == front))
        {
            cout << endl
```

```

        << "Queue is full";
    return;
}
//condition to make it circular
if (rear == MAX - 1)
    rear = 0;
else
    rear = rear + 1; //increasing index
arr[rear] = item; //adding element in queue
if (front == -1)
    front = 0;
}
//delq function definition
int delq()
{
    int data;
    //condition to check whether the queue is empty or not
    if (front == -1)
    {
        cout << endl
            << "Queue is empty";
        return 0;
    }
    else
    { //storing deleting element in data
        data = arr[front];
        if (front == rear)
        {

```

```

        front = -1;
        rear = -1;
    }
    else
    {
        if (front == MAX - 1)
            front = 0;
        else
            front = front + 1;
    }
    return data;
}
}
};

int main()
{
    queue a;
    a.addq(11);
    a.addq(12);
    a.addq(13);
    a.addq(14);
    a.addq(15);
    a.addq(16);
    a.addq(17);
    a.addq(18);
    a.addq(19);
    a.addq(20);

    //21 can not be added as queue gets full

```

```

a.addq(21);
//deleting elements
int i = a.delq();
cout << endl
    << "Item deleted=" << i;

a.addq(21); // data is saved at 0 location as its a circular Q
    //deleting elements
i = a.delq();
cout << endl
    << "Item deleted=" << i;
//deleting elements
i = a.delq();
cout << endl
    << "Item deleted=" << i;
return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

Queue is full
Item deleted=11
Item deleted=12
Item deleted=13

```

33. Write a program implanting linked list as a class. Also perform some required operation like inserting, deleting nodes & display the contents of entire linked list.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;
//linklist class definition
class linklist
{ //structure node definition
  struct node
  { //pointer and variable declaration
    int data;
    node *link;
  } * p; //pointer object declaration
  //public access specifier
public:
  //default constructor declaration
  linklist();
  //functions declaration for different purposes
  void append(int num);
  void addatbeg(int num);
  void addafter(int c, int num);
  void del(int num);
  void display();
  int count();
  //destructor declaration
  ~linklist();
};
//default constructor definition
linklist::linklist()
```

```

{
    p = NULL;
}
//append function definition
void linklist::append(int num)
{ //declaring pointer with node as datatype
    node *q,
      *t;
    //for first input
    if (p == NULL)
    { //creating new node with the help of "new"
        p = new node;
        //inserting inputted value in linklist
        p->data = num;
        //converting link to null
        p->link = NULL;
    }
    else
    {
        q = p;
        while (q->link != NULL)
            q = q->link;

        t = new node;
        t->data = num;
        t->link = NULL;
        q->link = t;
    }
}
}

```

```

//addatbeg function definition
void linklist::addatbeg(int num)
{
    node *q;
    q = new node;
    q->data = num;
    q->link = p;
    p = q;
}

//addafter function definition
void linklist::addafter(int c, int num)
{
    node *q, *t;
    int i;
    for (i = 0, q = p; i < c; i++)
    {
        q = q->link;
        if (q == NULL)
        {
            cout << endl
                << "There are less than " << c << "element";
            return;
        }
    }
    t = new node;
    t->data = num;
    t->link = q->link;
    q->link = t;
}

```



```

//del function definition
void linklist::del(int num)
{
    node *q, *r;
    q = p;
    if (q->data == num)
    {
        p = q->link;
        delete q;
        return;
    }
    r = q;
    while (q != NULL)
    {
        if (q->data == num)
        {
            r->link = q->link;
            delete q;
            return;
        }
        r = q;
        q = q->link;
    }
    cout << endl
         << "Element" << num << "not found";
}
//display function definition
void linklist::display()
{

```

```

node *q;
cout << endl;
for (q = p; q->link != NULL; q = q->link)
{
    cout << endl
        << q->data;
}
}
//count function definition
int linklist::count()
{
    node *q;
    int c = 0;
    for (q = p; q != NULL; q = q->link)
        c++;
    return (c);
}
//destructor definition
linklist::~~linklist()
{
    node *q;
    if (p == NULL)
        return;
    while (p != NULL)
    {
        q = p->link;
        delete p;
        p = q;
    }
}

```

```

}
int main()
{ //object declaration
  linklist ll;
  cout << endl
    << "No. of elements in linked list= " << ll.count();//calling count
  //calling append function
  ll.append(11);
  ll.append(22);
  ll.append(33);
  ll.append(44);
  ll.append(55);
  //calling addatbeg function
  ll.addatbeg(100);
  ll.addatbeg(200);
  ll.addatbeg(300);
  //calling addafter function
  ll.addafter(3, 333);
  ll.addafter(6, 444);
  //calling display function
  ll.display();
  cout << endl
    << "No. of element in linked list =" << ll.count();//calling count
  //calling del function
  ll.del(300);
  ll.del(66);
  ll.del(0);
  //calling display function
  ll.display();

```

```
cout << endl
    << "No. of element in linked list =" << ll.count();//calling count
return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

No. of elements in linked list= 0

300
200
100
11
333
22
33
444
44
No. of element in linked list =10
Element66not found
Element0not found

200
100
11
333
22
33
444
44
No. of element in linked list =9
```

34. Write a program implementing stack & its operations using dynamic memory allocation.

```
#include <iostream> // To include header file
using namespace std;

struct node // Structure of node
{
    int data;
    node *link; // Node type pointer link
};

class stack // To create class stack
{
private: // Private member top of node type pointer
    node *top;

public:
    stack() // Constructor initializes top to NULL
    {
        top = NULL;
    }
    void push(int item) // push function take one integer argument
    {
        node *temp; // Pointer declare of node type name temp
        temp = new node; // To allocate memory in heap
        if (temp == NULL) // It happens when Ram is full
            cout << endl
                << "Stack is full";
        temp->data = item; // To assign value of item in data part of temp
    }
};
```

```

temp->link = top; // To assign address of top in link part of temp
top = temp;      // To assign temp value in top
}
int
pop() // pop function, return integer argument
{
    if (top == NULL)
    {
        cout << endl
            << "Stack is empty";
        return 0;
    }
    node *temp;    // node type pointer temp declare
    int item;      // To declare item
    temp = top;    // To initialize temp with top
    item = temp->data; // To initialize item with data part of temp
    top = top->link; // To initialize top with link part of top
    delete temp;   // To delete temp node
    return item;   // To return the value of item
}
~stack() // Destructor, destroy total memory
{
    if (top == NULL) // If there is no element then return
        return;
    node *temp;
    while (top != NULL) // If there is element then delete it
    {
        temp = top;
        top = top->link;
    }
}

```

```

        delete temp;
    }
}
};
int main()
{
    stack s; // Object of class stack, s
    s.push(11); // push function call by s
    s.push(12);
    s.push(13);
    s.push(14);
    s.push(15);
    s.push(16);
    int i = s.pop(); // To Initialize i with return value of pop function
    cout << endl
        << "item popped=" << i; // To print value of i
    i = s.pop();
    cout << endl
        << "item popped=" << i;
    i = s.pop();
    cout << endl
        << "item popped=" << i;
    return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

item popped=16
item popped=15
item popped=14

```

35. Write a program implementing Queue stack & its operations using dynamic memory allocation.

CODE:

```
//Header file for standard input output
#include <iostream>
using namespace std;

//define structure node
struct node
{ //declaring variable and pointer for data and link
  int data;
  node *link;
};

//define class queue
class queue
{
private:
  node *front,
    *rear;

public:
  //default constructor
  queue()
  {
    front = rear = NULL;
  }
  //addq function definition to add element
  void addq(int item)
  {
```



```

node *temp;
temp = new node;
//condition to check whether the queue is full or not
if (temp == NULL)
    cout << endl
        << "Queue is full";
//inputting data in queue
temp->data = item;
temp->link = NULL;
if (front == NULL)
{
    rear = front = temp;
    return;
}
//switching node position for addition
rear->link = temp;
rear = rear->link;
}
//delq function definition to delete element
int delq()
{ //condition to check whether the queue is empty or not
    if (front == NULL)
    {

        cout << endl
            << "queue is empty";
        return 0;
    }
}
node *temp;

```

```

    int item;
    item = front->data;
    temp = front;
    front = front->link;
    delete temp;
    return item;
}
//Destructor definition
~queue()
{
    if (front == NULL)
        return;
    node *temp;
    while (front != NULL)
    {
        temp = front;
        front = front->link;
        delete temp;
    }
}
};
int main()
{ //object creation
    queue a;
    //adding element in queue
    a.addq(11);
    a.addq(12);
    a.addq(13);
    a.addq(14);
}

```

```
a.addq(15);
a.addq(16);
a.addq(17);
//deleting elements from queue
int i = a.delq();
cout << endl
    << "Item extracted=" << i;
i = a.delq();
cout << endl
    << "Item extracted=" << i;

i = a.delq();
cout << endl
    << "Item extracted=" << i;
return 0;
}
```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap

Item extracted=11
Item extracted=12
Item extracted=13
```

36. Write a program to implement the exception handling with multiple catch statements.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;

//test function definition
void test(int x)
{
    //try block of code starts
    try
    {
        //if the input is 1 then throw x i.e. integer 1
        if (x == 1)
            throw x;
        //else if the input is 0 then throw x i.e. character x
        else if (x == 0)
            throw 'x';
        //else if the input is -1 then throw x i.e. double 1.0
        else if (x == -1)
            throw 1.0;
        cout << "End of try-block\n";
    }
    //catch block starts
    catch (char c) //for char
    {
        cout << "Caught a Character\n";
    }
    catch (int c) //for int
```

```

{
    cout << "Caught an Integer\n";
}
catch (double c) //for double
{
    cout << "Caught a Double\n";
}
cout << "End of try-catch system\n";
}
int main()
{
    cout << "Testing Multiple Catches\n";
    cout << "x==1\n";
    //calling test function with input 1
    test(1);
    cout << "x==0\n";
    //calling test function with input 0
    test(0);

    cout << "x==2\n";
    //calling test function with input 2
    test(2);
    return 0;
}

```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Testing Multiple Catches
x==1
Caught an Integer
End of try-catch system
x==0
Caught a Character
End of try-catch system
x==2
End of try-black
End of try-catch system
```

37. Write a program to implement the exception handling with rethrowing in exception.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;

//divide function definition with params x and y of type double
void divide(double x, double y)
{

    cout << "Inside Function\n";
    //try block starts
    try
    { //if the input y is 0.0 then throw the y
        if (y == 0.0)
            throw y;
        //else print output of division
        else
            cout << "Division =" << x / y << "\n";
    }
    //catch block starts
    catch (double)
    {
        cout << "Caught double inside function\n";
        //throwing the double to main
        throw;
    }
    cout << "End of Function\n";
}
```

```

int main()
{
    cout << "Inside Main\n";
    //try block starts
    try
    {

        //calling divide function
        divide(10.5, 2.0);
        //calling divide function
        divide(20.0, 0.0);
    }
    //catch block starts
    catch (double)
    { //printing output
        cout << "Caught double inside main\n";
    }
    cout << "End of Main\n";
    return 0;
}

```

OUTPUT:

```

PS C:\Users\HARSH\OneDrive\Documents\cpp> g++ hem.cpp -o ap
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Inside Main
Inside Function
Division =5.25
End of Function
Inside Function
Caught double inside function
Caught double inside main
End of Main

```



38. Write a program to implement the exception handling with the functionality of testing the throw restrictions.

CODE:

```
//Header file for standard input and output
#include <iostream>
using namespace std;

//test function definition with restriction of any other argument type instead of int and double
void test(int x) throw(int, double)
{
    //if input x=0 then throw character x
    if (x == 0)
        throw 'x';
    //else if input x==1 throw integer x
    else if (x == 1)
        throw x;
    //else if input x==1 throw double 1.0
    else if (x == -1)
        throw 1.0;
    cout << "End of Function Block\n";
}

int main()
{
    //try block starts
    try
    {
        cout << "Testing Throw Restrictions\n";
        cout << "x == 0\n";
        //calling test for value 0
```

```

test(0);
cout << "x == 1\n";
//calling test for value 1
test(1);
cout << "x == -1\n";
//calling test for value -1
test(-1);
cout << "x == 2\n";
//calling test for value 2
test(2);
}
//catch block starts
catch (char c) //for character
{
    cout << "Caught a Character\n";
}
catch (int m) //for integer
{
    cout << "Caught an Integer\n";
}

catch (double d) //for double
{
    cout << "Caught a Double\n";
}
cout << "End of Try-catch system\n";
return 0;
}

```

OUTPUT:

```
PS C:\Users\HARSH\OneDrive\Documents\cpp> ./ap
Testing Throw Restrictions
x == 0
terminate called after throwing an instance of 'char'
PS C:\Users\HARSH\OneDrive\Documents\cpp> █
```