UNIT-1

2019-20

1. (a)What is the difference between Procedural Programming and Object-oriented programming? Explain with examples.

ANS.

PROCEDURAL ORIENTED PROGRAMMING	OBJECT ORIENTED PROGRAMMING
In procedural programming, the program is divided into small parts called ***functions***.	In object-oriented programming, the program is divided into small parts called ***objects***.
Procedural programming follows a ***top-down approach***.	Object-oriented programming follows a ***bottom-up approach***.
There is no access specifier in procedural programming.	Object-oriented programming has access specifiers like private, public, protected, etc.
Adding new data and functions is not easy.	Adding new data and function is easy.

Procedural programming does not have any proper way of hiding data so it is ***less secure***.	Object-oriented programming provides data hiding so it is ***more secure***.
In procedural programming, overloading is not possible.	Overloading is possible in object-oriented programming.
In procedural programming, the function is more important than data.	In object-oriented programming, data is more important than function.
Procedural programming is based on an ***unreal world***.	Object-oriented programming is based on the ***real world***.
Examples: C, FORTRAN, Pascal, Basic, etc.	Examples: C++, Java, Python, C#, etc.

(b)Define a class named Admission in C++ with the following description.
Private members:

- **admno -integer(Ranges 10-1500)**
- **Name - string of 20 characters**
- **Cls -integer**
- **Fees -float**

Public Members:

- **A constructor which initialized admno with 10, name with “NULL”, cls with 0, and fees with 0**
- **Function getdata() to read the object of Admission type**
- **Function putdata() to print the details of objects of admission type.**
- **Function draw_nos() to generate the admission no. randomly to match with admno and display the details of an object.**

ANS.

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

class admission
{
  int admno;
  char name[20];
  int cls;
  float fees;

public:
  admission()
  {
    char name = 'h';
    admno = 10;

    fees = 0;
    cls = 0;
  }
  void getdata();
  void putdata();
  void draw_nos();
};
void admission ::getdata()
{
  cout << "\n Enter admission No";
  cin >> admno;
  cout << "\n Enter Name";
```

```
 gets(name);
 cout << "\n Enter class";
 cin >> cls;
 cout << "\n Enter Fees";
 cin >> fees;
}
void admission ::putdata()
{
 cout << "\n Admission No :" << admno;
 cout << "\n Name :";
 puts(name);
 cout << "\n Class";
 cout << cls;
 cout << "\n Fees :" << fees;
}
void admission ::draw_nos()
{
 int adno;
 adno = rand();
 if (admno == adno)
 {
  putdata();
 }
}
int main()
{
 admission o1;
 o1.putdata();
 return 0;
}
```

2. (a) Explain the properties of OOPS in detail with examples.

ANS. GFG LINK OF THE ANSWERS

(b) Demonstrate the structure of the CPP program with the help of examples.

ANSWER LINK-

2018-19

1. **(a) Explain Object-Oriented Programming and its concepts. ANS**

(b) What is the difference between while and do-while loop? ANS

2. **(a) Define Classes and Objects. ANS**

(b) Explain the concept of static and global variables with examples. ANS

2017-18

1. **(a)What is class and object? Explain with examples?ANS**

(b)Describe Data Abstraction and Encapsulation with an example? ANS EXAMPLE

3. **(a) Explain the access modifier in C++ with an example. ANS**

(b) Briefly explain the following:-

(i)Multiple Inheritance.ANS

(ii) Scope Resolution Operator ANS

2016-17

1.(a) What are the important features of OOP language? Explain data encapsulation in detail. FEATURES-GFG LINK OF THE ANSWERS DATA ENCAPSULATION---ANS EXAMPLE

(b)Write a program in c++ to calculate the LCM of any two non-zero positive integer numbers. ANS

2. (a)Explain the call by value and call by reference parameter passing with suitable examples. ANS

(b). What is the inline function? What are the limitations of its? Explain, how it is different from Macros?

ANS.

The inline function is the optimization technique used by the compilers. One can simply prepend an inline keyword to a function prototype to make a function inline. Inline function instruct compiler to insert complete body of the function wherever that function got used in code.

Advantages:- 1) It does not require function calling overhead.

2) It also saves the overhead of variables push/pop on the stack, while function calling.
3) It also saves the overhead of return call from a function.
4) It increases the locality of reference by utilizing an instruction cache.
5) After in-lining compiler can also apply interprocedural optimization if specified. This is the most important one, in this way compiler can now focus on dead code elimination, can give more stress on branch prediction, induction variable elimination, etc.

Disadvantages:-
1) May increase function size so that it may not fit on the cache, causing lots of cache misses.
2) After the in-lining function if the variables number which is going to use the register increases then they may create overhead on the register variable resource utilization.
3) It may cause compilation overhead as if somebody changes code inside an inline function then all calling locations will also be compiled.
4) If used in the header file, it will make your header file size large and may also make it unreadable.
5) If somebody used too many inline functions resultant in a larger code size then it may cause thrashing in memory. More and more page faults bringing down your program performance.
6) It's not useful for embedded systems where large binary size is not preferred at all due to memory size constraints.

DIFFERENCE

In solving the C++ problem of a macro with access to private class members, *all* the problems associated with preprocessor macros were eliminated. This was done by bringing the concept of macros under the control of the compiler where they belong. C++ implements the macro as an *inline function*, which is a true function in every sense. Any behavior you expect from an ordinary function, you get from an inline function. The only difference is that an inline function is expanded in place, like a preprocessor macro, so the overhead of the function call is eliminated. Thus, you should (almost) never use macros, only inline functions.

Any function defined within a class body is automatically inline, but you can also make a non-class function inline by preceding it with the inline keyword. However, for it to have any effect, you must include the function body with the declaration, otherwise, the compiler will treat it as an ordinary function declaration. Thus,

2015-16

1. (a)What is the use of the Preprocessor Directive? Explain with an example?
ANS

As the name suggests Preprocessors are programs that process our source code before compilation. There are a number of steps involved between writing a program and executing a program in C / C++.
Preprocessor programs provide preprocessors directives that tell the compiler to preprocess the source code before compiling. All of these preprocessor directives begin with a '#' (hash) symbol. The '#' symbol indicates that, whatever statement starts with #, is going to the preprocessor program, and the preprocessor program will execute this statement. Examples of some preprocessor directives are: *#include*, *#define*, *#ifndef* etc. Remember that # symbol only provides a path that it will go to the preprocessor, and command such as include is processed by the preprocessor program. For example, include will include extra code to your program. We can place these preprocessor directives anywhere in our program.
There are 4 main types of preprocessor directives:

1. Macros
2. File Inclusion
3. Conditional Compilation
4. Other directives

```
#include <iostream>//for input and output

// macro with parameter
#define AREA(l, b) (l * b)//macros
int main()
{
    int l1 = 10, l2 = 5, area;

    area = AREA(l1, l2);

    printf("Area of rectangle is: %d", area);

    return 0;
}
```

(b)What is the use of the inline function? Give code segment which shows the use of inline functions?

ANS.

USE

In many places, we create the functions for small work/functionality which contain simple and less number of executable instruction. Imagine their calling overhead each time they are being called by callers.

When a normal function call instruction is encountered, the program stores the memory address of the instructions immediately following the function call statement, loads the function being called into the memory, copies argument values, jumps to the memory location of the called function, executes the function codes, stores the return value of the function, and then jumps back to the address of the instruction that was saved just before executing the called function. Too much run time overhead.

The C++ inline function provides an alternative. With an inline keyword, the compiler replaces the function call statement with the function code itself (a process called expansion) and then compiles the entire code. Thus, with inline functions, the compiler does not have to jump to another location to execute the function, and then jump back as the code of the called function is already available to the calling program.

With the below pros, cons, and performance analysis, you will be able to understand the “why” for inline keyword

Pros:-

1. It speeds up your program by avoiding function calling overhead.
2. It saves the overhead of variables push/pop on the stack when function calling happens.
3. It saves the overhead of return call from a function.
4. It increases the locality of reference by utilizing an instruction cache.
5. By marking it as inline, you can put a function definition in a header file (i.e. it can be included in multiple compilation units, without the linker complaining)

```
#include <iostream>
using namespace std;
inline int max(int a, int b)
{
    return a > b ? a : b;
}
int main()
{
    cout << max(10, 20);         // == cout << (10>20 ? 10 : 20)
    cout << " " << max(99, 88); // == cout << " " << (99>88 ? 99 : 88)
    return 0;
}
```

2.(a) Define automatic,external,static variable in c++.

(b) Write a c++ program to print the largest even and largest odd number from a list of numbers entered through the user. The list terminates as soon as one enters 0.

```
# include <iostream>

using namespace std;

int main()
{
      int num, meven=0, modd=1;

      cout<<"ENTER NUMBER (0 TO TERMINATE):\n";

      do
      {
          cin>>num;

           if(num%2==0)
          {
                if(num>meven)
                meven=num;
          }
          else
          {
                if(num>modd)
                modd=num;
          }

      }while(num);

       cout<<"LARGEST EVEN NUMBER IS:"<<meven<<endl;
       cout<<"LARGEST ODD NUMBER IS:"<<modd<<endl;

       return 0;
}
```

2014-15

1. **(a)What is object-oriented programming? Explaining the primitive data types in c++.**

ANS:
OOPS-ABOVE:
PRIMITIVE:

A primitive type is a data type where the values that it can represent have a very simple nature (a number, a character or a truth-value); the primitive types are the most basic building blocks for any programming language and are the base for more complex data types.

C++ has the following primitive data types −

S.No	Type	Description
1	bool	Stores either value true or false.
2	char	Typically a single octet (one byte). This is an integer type.
3	int	The most natural size of an integer for the machine.
4	float	A single-precision floating-point value.
5	double	A double-precision floating-point value.
6	void	Represents the absence of type.

(b)When will you make a function inline? How does an inline function differ from a preprocessor macro? Explain with an example.
(c) Compare break, continue, goto, and exit statements **ANS---**

2.Write short notes (any four):

- **Structure of the C++ program**
- **Storage classes**
- **Defining member function**
- **Default arguments**
- **Classes and Objects.**

UNIT-2

2019-20

3. (a)What do you understand by the constructor. Give the difference between Constructor and function. ANS
(b) Explain the binary Operator Overloading in C++. List the operation that cannot be overload.
ANS

List the operation that cannot be overload.

```
1> Scope Resolution Operator  (::)
2> Pointer-to-member Operator (.*)
3> Member Access or Dot operator  (.)
4> Ternary or Conditional Operator (?:)
5> Object size Operator   (sizeof)
6> Object type Operator   (typeid)
```

4. (a)In CPP, how we can allocate dynamic memory? Explain with the example. ANS
(b). Explain Pointers to the class example. Write a program in CPP to compare two strings using pointers.
ANS 1. ANS2.

2018-19

3. (a) Define pointer and write a C++ program to explain the concept of the pointer. ANS

4. (a)Write a c++ program to demonstrate the function of the array.
(b)Define Operator overloading with an example. ANS

2017-18

3. (a)Explain the various control structures available in c++. ANS
(b) How does C-in and Cout works with >> and << operator.
4. (a)Differentiate between User Defined and library FUnction with an example. ANS
(b)Describe a function Passing Parameter Call by Value and by Reference. ANS

2016-17

3. (a)What is the difference between constructors and destructors? ANS Explain copy constructors with suitable examples. ANS

(b)What is operator overloading? ANS Write a c++ program to overload + operator for the addition of two matrices.

ANS

```
#include <iostream.h>
#include <iomanip.h>
class matrix
{
    int maxrow, maxcol;
    int *ptr;

public:
    matrix(int r, int c)
    {
        maxrow = r;
        maxcol = c;
        ptr = new int[r * c];
    }
    void getmat()
    {
        int i, j, mat_off, temp;
        cout << endl
             << "enter elements matrix:" << endl;
        for (i = 0; i < maxrow; i++)
        {
            for (j = 0; j < maxcol; j++)
            {
                mat_off = i * maxcol + j;
                cin >> ptr[mat_off];
            }
        }
    }
    void printmat()
    {
        int i, j, mat_off;
        for (i = 0; i < maxrow; i++)
        {
            cout << endl;
```

```
            for (j = 0; j < maxcol; j++)
            {
                mat_off = i * maxcol + j;
                cout << setw(3) << ptr[mat_off];
            }
        }
    }
    int delmat()
    {
        matrix q(maxrow - 1, maxcol - 1);
        int sign = 1, sum = 0, i, j, k, count;
        int newsize, newpos, pos, order;
        order = maxrow;
        if (order == 1)
        {
            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++;
}
            }
            sum = sum + ptr[i] * sign * q.delmat();
        }
        return (sum);
    }
    matrix operator+(matrix b)
    {
        matrix c(maxrow, maxcol);
```

```
        int i, j, mat_off;
        for (i = 0; i < maxrow; i++)
        {
            for (j = 0; j < maxcol; j++)
            {
                mat_off = i * maxcol + j;
                c.ptr[mat_off] = ptr[mat_off] + b.ptr[mat_off];
            }
        }
        return (c);
    }
    matrix operator*(matrix b)
    {
        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++)
            {
                mat_off3 - i *c.maxcol + j;
                c.ptr[mat_off3] = 0;
                for (k = 0; k < b.maxrow; k++)
                {
                    mat_off2 = k * b.maxcol + j;
                    mat_off1 = i * maxcol + k;
                    c.ptr[mat_off3] += ptr[mat_off1] * b.ptr[mat_off2];
                }
            }
        }
        return (c);
    }
    int operator==(matrix b)
    {
        int i, j, mat_off;
        if (maxrow != b.maxrow || maxcol != b.maxcol)
            return (0);
        for (i = 0; i < maxrow; i++)
        {
            for (j = 0; j < maxcol; j++)
            {
```

```
                mat_off = i * maxcol + j;
 if( ptr[ mat_off ]

 != b.ptr[ mat_off ] )
     return (0);
            }
        }
        return (1);
    }
};
void main()
{
    int rowa, cola, rowb, colb;
    cout << endl
         << "Enter dimensions of matrix A ";
    cin >> rowa >> cola;
    matrix a(rowa, cola);
    a.getmat();
    cout << endl
         << "Enter dimensions of matrix B";
    cin >> rowb >> colb;
    matrix b(rowb, colb);
    b.getmat();
    matrix c(rowa, cola);
    c = a + b;
    cout << endl
         << "The sum of two matrics = ";
    c.printmat();
    matrix d(rowa, colb);
    d = a * b;
    cout << endl
         << "The product of two matrics = ";
    d.printmat();
    cout << endl
         << "Determinant of matrix a =" << a.delmat();
    if (a == b)
        cout << endl
             << "a & b are equal";
    else
        cout << endl
```

```
            << "a & b are not equal";
}
```

4. (a)What is the use of 'this' pointer? ANS Why 'new' is better than 'malloc' for dynamic memory allocation? Explain. ANS
(b)Write a program to create a linked list of objects. Also, define the functionality for the insertion and deletion of objects in the linked list. FROM FILE

2015-16

3. (a)What is the difference between calling methods for constructors and destructor. ANS

(b)Write a c++ program to create an array of strings. Read and display the string using constructor and destructor. Do not use the member function.

4. (a) List the difference between operator overloading and function overloading? ANS EXTRA
(b) Write a C++ program to create dynamically an array of objects of class type. Use a new operator? ANS

2014-15

3. (a)What is a constructor? List some of the special properties of the constructor function. Also, give an example of a copy constructor.
(b)How an object may be used as a function argument? Explain with an example. ANS

4. (a)What is operator overloading? Define a class Strings. use overloaded == operator to compare two strings. ANS
(b)What is a linked list? Explain and implement functionality for insertion and deletion of a node in a linked list using c++. ANS 1 PROGRAM FROM FILE

UNIT-3

2019-20

5. (a)Explain different types of inheritance with help of examples. ANS
(b) Differentiate between function overloading and function overridden. ANS

6.(a)Explain the working of different types of visibility mode in inheritance. ANS
(b)Explain Run time Polymorphism with its advantages. Explain virtual functions with their needs, properties, and examples. ANS

2018-19

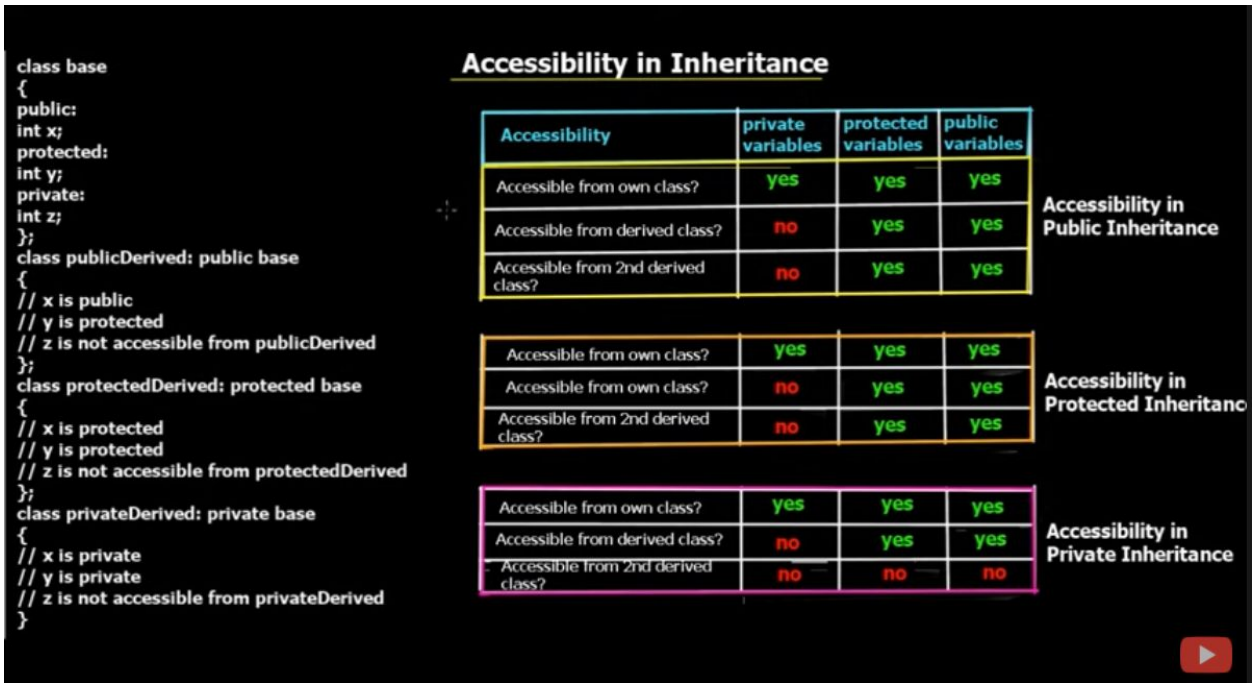
5. (a)Explain the concept of Inheritance with its types. ANS
(b)What is the use of the static function. Explain? ANS
6. (a)What do you understand by virtual function and friend function? VIRTUAL FRIEND
(b)Write a c++ program that shows an overriding member function.ANS

2017-18

5. (a)Explain Static and Extern Variable with an example. STATIC EXTERN
(b). Explain Function Overloading and Constructor Overloading.

2016-17

5.(a)Differentiate the Multiple, Multi-level, and Hierarchical inheritance. How does an object of a derived class, access the members of the base class?
ANS



(b)Explain the following with the help of an example:
(i)Friend Function ANS
(ii)Static member Function. ANS

6. (a)Explain Run time polymorphism and its advantages. How is it implements in c++? ANS
Also, discuss abstract class and its purpose.

2015-16

5. (a)What is dynamic binding? How it is perform?
(b)Write a c++ program to create dynamically an array of objects of class type. Use a new operator? ANS

6. (a)Explain the use of friend function? ANS
(b)What is the Overriding member function? Explain. ANS

2014-15

5. (a)What does inheritance mean in c++?What are the different forms of inheritance? Give example for each. ANS
(b) Explain 'derived class constructor' with suitable examples. ANS
6. (a)What is a friend function? What are the merits and demerits of using a friend function? Give an example of the friend function. ANS
(b)What is a virtual function? Why do we need a virtual function? Explain with an example. VIRTUAL

UNIT-4

2019-20

7. (a)What do you understand by template programming. Explain with the help of an example. ANS
(b)Write a program to overload cin and cout operators. ANS /FILE

8. (a)Briefly explain the concept of Exception Handling in CPP. ANS
(b)Write a CPP program to count the total number of words, lines, and the total size of a Text file. ANS

2018-19

7.(a)Explain LiskedList using Templates.

```
#include

using namespace std;

template class LinkedList
{
private:
    struct Node
    {
        struct Node *prev;
        struct Node *next;
        T data;
    };
    struct Node *pNode;

public:
    LinkedList();
    ~LinkedList();
    void append(T);
    void addAtBeg(T);
    void addAfter(int, T);
    void deleteNode(T);
    void display();
    void count();
};

template LinkedList::LinkedList()
{
    pNode = NULL;
}

template LinkedList::~LinkedList()
{
    struct Node *temp = pNode;

    while (pNode)
    {
        temp = pNode;
```

```
        pNode = pNode->next;
        delete temp;
    }
}

template void LinkedList::append(T data)
{
    if (pNode == NULL)
    {
        pNode = new struct Node;
        pNode->next = NULL;
        pNode->prev = NULL;
        pNode->data = data;
        return;
    }
    else
    {
        struct Node *temp = pNode;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = new struct Node;
        temp->next->prev = temp;
        temp->next->next = NULL;
        temp->next->data = data;
    }
}
template void LinkedList::addAtBeg(T data)
{

    if (pNode == NULL)
    {
        pNode = new struct Node;
        pNode->next = NULL;
        pNode->prev = NULL;
        pNode->data = data;
        return;
    }
    else
```

```
    {
        struct Node *temp = new struct Node;
        temp->next = pNode;
        pNode->prev = temp;
        temp->prev = NULL;
        temp->data = data;
        pNode = temp;
    }
}

template void LinkedList::addAfter(int position, T data)
{
    cout << "Assumed will have some element and position is a valid
position" < next;
}
struct Node *tempNode = new struct Node;
tempNode->data = data;
tempNode->prev = temp;
tempNode->next = temp->next;
temp->next = tempNode;
}

template void LinkedList::deleteNode(T data)
{
cout << "Here also not handled the condition, assumed data is in the
link"<data != data)
{
    temp = temp->next;
}
temp->prev->next = temp->next;
temp->next->prev = temp->prev;
delete temp;
}

template void LinkedList::display()
{
    cout << "Not checked all the condition,assumed that list have some
element" < next;
}
cout << "Number of elements in the list = " << count << endl;
```

```
}

int main()
{

    LinkedList *pLinkObj = new LinkedList();

    pLinkObj->append(2);
    pLinkObj->append(4);
    pLinkObj->append(6);
    pLinkObj->append(8);
    pLinkObj->append(10);

    cout << "======" << endl;
    pLinkObj->display();
    cout << "======" < addAtBeg(1);
    cout << "======" << endl;
    pLinkObj->display();
    cout << "======" < addAfter(4, 7);
    cout << "======" << endl;
    pLinkObj->display();
    cout << "======" < deleteNode(8);
    pLinkObj->display();

    pLinkObj->count();

    delete pLinkObj;
    return 0;
}
```

(b)What is the purpose of Templates in C++.
ANS

8. (a)Explain Try, Catch, and throw in brief with respect to Execution Handling. ANS

(b)Write a c++ program to create your own exception. ANS

2017-18

7. (a)Discuss all different types of inheritance C++ supports. ANS

(b)What is Virtual Function? How does it work? ANS

8.Write Short notes(Any two)-

- **Input-Output flags ANS**
- **Pure virtual function. ANS**
- **Pointer to objects. ANS**

2016-17

7. (a)Explain file pointers and multiple programs in c++. ANS
(b) What is an exception? Explain exceptions with arguments. ANS

8. (a)What are templates? Write a template function search () for searching an item in arrays of integers as well as float numbers. ANS
(b) Write a short note on the following:
(i)Istream Class ANS
(ii)Standard Template library. ANS

2015-16

7. (a)Write a program in c++ to write the contents of one file in reverse order into another file. ANS
(b)Explain various file mode exist in c++. ANS

8. (a)What is a template class. How we can create a linked list using a template class.
(b)Write a short note on the following:
(i) Standard Template Library ANS
(ii)Stream Errors ANS

2014-15

7. (a)What is a stream? Describe the features of the I/O system supported by c++. ANS
(b)How is the cin and cout operators can be overloaded? Implement this using a C++ program. ANS
(c)Write a short note on multi-file programs and projects. ANS

8. (a)Write a function template for finding the minimum value contained in an array. ANS
(b)What is an exception? How is an exception handled in c++? Write a program containing a possible exception. Use a try block to throw it and a catch block to handle it properly.
ANS