COMPUTER SCIENCE WITH C++

FOR CBSE CLASS 12

Revision Notes – 2013

Part -I

[Structure, Class and Object, Constructor and Destructor, Inheritance, File Handling, Pointer, Array, Data Structure and class 11 th revision tour]

(Question 1,2,3,4 of Board Exam Paper)

CONTENTS

STRUCTURE

A structure is a collection of variable which can be same or differ ent types. You can refer to a structure as a single variable, and to its parts as **members** of that variable by using the dot (.) operator. The power of structures lies in the fact that once defined, the structure name becomes a **user-defined data type** and may be used the same way as other built-in data types, such as int, double, char.

```
struct   STUDENT
{
int rollno, age;
char name[80];
float marks;
} ;

void main()
{

STUDENT s1, s3;            // declare two variables of the new type
cin>>s1.rollno>>s1.age>>s1.name>>s1.marks;          //accessing of data members
cout<<s1.rollno<<s1.age<<s1.name<<s1.marks;
STUDENT s2 = {100,17,”Aniket”,92};        //initialization of structure variable
cout<<s2.rollno<<s2.age<<s2.name<<s2.marks;
s3=s2;                  //structure variable in assignment statement
cout<<s3.rollno<<s3.age<<s3.name<<s3.marks;
}
```

Defining a structure

When dealing with the students in a school, many variables of dif ferent types are needed. It may be necessary to keep track of name, age, Rollno, and marks point for example.

```
struct STUDENT
{
int rollno, age;
char name[80];
float marks;
};
```

STUDENT is called the **structure tag,** and is your brand new data type, like int, double or char. **rollno, name, age,** and **marks** are **structure members.**

Declaring Variables of Type struct

The most efficient method of dealing with structure variables is to define the structure **globally**. This tells "the whole world", namely main and any functions in the program, that a new data type exists. To declare a structure globally, place it **BEFORE** void main(). The structure variables can then be defined locally in main, for example…

```
struct STUDENT
{
int rollno, age;
char name[80];
float marks;
};

void main()
{
// declare two variables of the new type
STUDENT s1, s3;
………
}
```

Alternate method of declaring variables of type struct:

```
struct STUDENT
{
int rollno, age;
char name[80];
float marks;
} s1, s3;
```

Accessing of data members

The accessing of data members is done by using the following format:
structure variable.member name
for example
cin>>s1.rollno>>s1.age>>s1.name>>s1.marks;

Initialization of structure variable

Initialization is done at the time of declaration of a variable. For example
STUDENT s2 = {100,17,"Aniket",92};

Structure variable in assignment statement
The statement
s3=s2;
assigns the value of each member of s2 to the corr esponding member of s3. Note that one structure variable can be assigned to another only when they ar e of the same structure type, otherwise complier will give an error.

Nested structure (St ructure within structure)

It is possible to use a structure to define another structure. This is called nesting of structure. consider the following program

```
struct DAY
{
int month, date, year;
};

struct STUDENT
{
int rollno, age;
char name[80];
day date_of_birth;
float marks;
};
```

typedef

It is used to define new data type for an existing data type. It provides and alternative name for standard data type. It is used for self documenting the code by allowing descriptive name for the standard data type.

The general format is:

typedef *existing datatype new datatype*

for example:
typedef float real;

Now, in a program one can use datatype real instead of float.
Therefore, the following statement is valid:
real amount;

Enumerated data type

The enum specifier defines the set of names which are stored internally as integer constant. The first name was given the integer value 0, the second value 1 and so on.

for example:
enum months{jan, feb, mar, apr, may} ;

It has the following features:

- It is user defined.
- It works if you know in advance a finite list of values that a data type can take.
- The list cannot be input by the user or output on the screen.

#define preprocessor directive

The #define preprocessor allows to define symbolic names and constants e.g.
#define pi 3.14159
This statement will translate every occurrence of PI in the program to 3.14159

Macros

Macros are built on the #define preprocessor. Normally a macro would look like:
#define square(x) x*x
Its arguments substituted for replacement text, when the macro is expanded.

ASSIGNME NT

QUESTION 1.
Give the output of the following program

```
#include<iostream.h>
struct Pixel
{
int C,R;
};
void Display(Pixel P)
{
cout<<"Col”<<P.C<<"Row"<<P.R<<endl;
}
void main()
{
Pixel X={40,50},Y,Z;
Z=X;
X.C+=10;
Y=Z;
Y.C+=10;
Y.R+=20;
Z.C-=15;
Display(X);
Display(Y);
Display(Z);
}
```

OOP CONCEPTS

Paradigm -: It means organizing principle of a pr ogram. It is an approach to programming.
Procedural Paradigm
In PROCEDURAL PROGRAMMING PARADIGM, the emphasis is on doing things i.e., the procedure or the algorithm. The data takes the back seat in procedural programming paradigm. Also, this paradigm does not model real world well.
Object Orient ed programm ing
The OBJECT ORIENTED PROGRAMMING PARADIGM models the real world well and overcomes the shortcomings of procedural paradigm. It views a problem in terms of objects and thus emphasizes on both procedures as well as data.

The following are the **basic concepts used in object-oriented programming** .
Object- : An object is an identifiable entity with some characteristics and behavior.
Class- : A class represents a group of objects that share common properties, behavior and relationships.
Data Abst raction- : Abstraction refers to act of representing essential features without including the background details or explanations.
Encapsulation- : The wrapping up of data and associated functions into a single unit is known as ENCAPSULATION. Encapsulation implements data abstraction.
Inherit ance- : It is the capability of one class of things to inherit capabilities or properties from another class.
Polymorphism- : It is the ability for a message or data to be processed in more than one form. Polymorphism is a property by which the same message can be sent to objects of several different classes. Polymorphism is implemented in C++ through virtual functions and overloading- function overloading and operator overloading.

Advantages of Object orient ed programm ing.
Software complexity can be easily managed
Object-oriented systems can be easily upgraded
It is quite easy to partition the work in a project based on object

class enforce dat a-hiding, abstraction & encapsulation
A class groups its members into three sections : private, protected, and public.
The private and protected members remain hidden from outside world. Thus through private and protected members, a class enforces data-hiding.
The outside world is given only the essential and necessary information through public members, rest of the things remain hidden, which is nothing but abstraction. Abstraction means representation of essential features without including the background details and explanation.

CLA SS ES & O BJ ECTS

A CLASS is an expanded concept of a data structure: instead of holding only data, it can hold both data and functions. An OBJECT is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable. Classes are generally declared using the keyword class, with the following format:

```
class class_name
{
access_specifier_1:
member1;
access_specifier_2:
member2;
...
}object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members, that can be either data or function declarations, and optionally access specifiers.

An access specifier is one of the following three keywords: private, public or protected. These specifiers modify the access rights that the members following them acquire:

private m embers of a class are accessible only from within other members of the same class or from their FRIENDS.

protected members are accessible from members of their same class and from their friends, but also from members of their derived classes.

Finally, **public members** are accessible from anywhere where the object is visible.

By default, all members of a class declared with the class keyword have pr ivate access for all its members. Therefore, any member that is declared before one other class specifier automatically has private access.

AS SI GNM ENT

QUESTION 2

Define a class student with the following specification

Private members of class student

admno	integer
sname	20 character
eng. math, science	float
total	float
ctotal()	a function to calculate eng + math + science with float return type.

Public m ember function of class student

Takedata()	Function to accept values for admno, sname, eng, science and invoke ctotal() to calculate total.
Showdata()	Function to display all the data members on the screen.

CON S TRUCT OR A N D D ES T RUCT OR

CONSTRUCTOR

It is a member function having same name as it s class and which is used to initialize the objects of that class type with a legal initial value. Constructor is automatically called when object is created.

Types of Constructor

Def ault Constructor-: A constructor that accepts no parameters is known as default constructor. If no constructor is defined then the compiler supplies a default constructor.

```
student :: student()
{
rollno=0;
}
```

Paramet erized Constructor - : A constructor that receives arguments/parameters, is called parameterized constructor.

```
student :: student(int r)
{
rollno=r;
}
```

Copy Const ructor- : A constructor that initializes an object using values of another object passed to it as parameter, is called copy constructor. It creates the copy of the passed object.

```
student :: student(student &s)
{
rollno = s.rollno;
}
```

There can be multiple constructors of the same class, provided they have different signatures. This feature is called constructor overloading.

DESTRUCTOR

A destructor is a member function having sane name as that of its class preceded by ~(tilde) sign and which is used to destroy the objects that have been cr eated by a constructor. It gets invoked when an object s scope is over.

```
~student() { }
```

QUESTION 3

Answer the questions (i) and (ii) after going through the following class:

```
class Exam
{
int Marks;
char Subject[20];
public:
Exam ()                                          //Function 1
{
Marks = 0;
strcpy (Subject,”Computer”);
}
Exam(char S[])                              //Function 2
{
Marks = 0;
strcpy(Subject,S);
}
Exam(int M)                                    //Function 3
{
Marks = M;
strcpy(Subject,”Computer”);
}
Exam(char S[], int M)                   //Function 4
{
Marks = M;
strcpy (Subject,S);
}

Exam(Exam &E);      //Function 5

~Exam()        //Function 6
{}
};
```

(i) Write statements in C++ that would execute Function 3 and Function 4 of class Exam.

(ii) Which feature of Object Oriented Programming is demonstrated using Function 1, Function 2, Function 3 and Function 4 in the above class Exam?

iii) In Object Oriented Progr amming, what is Function 6 referred as and when does it get invoked/called?

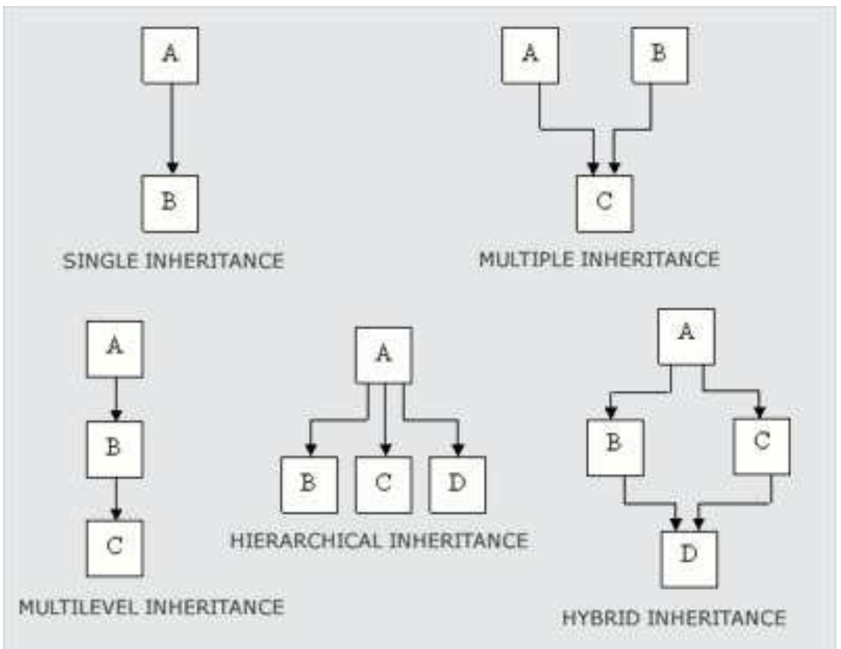
iv) Which category of constructor - Function 5 belongs to? Write complete definition of it.

I N HE RI TA NCE

Inherit ance: It is the capability of one class to inherit properties from another class.
Base Class: It is the class whose properties are inherited by another class. It is also called Super Class.
Derived Class: It is the class that inherit properties from base class(es).It is also called Sub Class.

FORMS OF INHERITANCE

Single Inherit ance: It is the inheritance hierarchy wherein one derived class inherits from one base class.
Multiple Inheritance: It is the inheritance hierarchy wherein one derived class inherits from multiple base class(es)
Hierarchical Inheritan ce: It is the inheritance hierar chy wherein multiple subclasses inherits from one base class.
Multilevel Inheritance: It is the inheritance hierarchy wherein subclass acts as a base class for other classes.
Hybrid Inherit ance: The inheritance hierarchy that reflects any legal combination of other four types of inheritance.



Visibility Mode: It is the keyword that controls the visibility and availability of inherited base class members in the derived class.It can be either private or protected or public.

Private Inheritance: It is the inheritance facilitated by private visibility mode. **In private inheritance ,the protected and public members of base class become private members of the derived class.**

Public Inheritance: It is the inheritance facilitated by public visibility mode. **In public inheritance ,the protected members of base class become protected members of the derived class and public members of the base class become public members of derived class.**

Protected Inheritance: It is the inheritance facilitated by protected visibility mode. **In protected inheritance ,the protected and public members of base class become protected members of the derived class.**

AS SI GNM ENT

QUESTION 4

Answer the questions (i) to (iv) based on the following:

```
class PUBLISHER
{
char Pub[12];
double Turnover;
protected:
void Register();
public:
PUBLISHER();
void Enter();
void Display();
};
class BRANCH
{
char CITY[20];
protected:
float Employees;
public:
BRANCH();
void Haveit();
void Giveit();
};
class AUTHOR : private BRANCH, public PUBLISHER
{
int Acode;
char Aname[20];
float Amount;
public:
AUTHOR();
void Start();
void Show();
};
```

i) Write the names of data members, which are accessible from objects belonging to class AUTHOR.

ii) Write the names of all the member functions which are accessible from objects belonging to class AUTHOR.
iii) Write the names of all the members which are accessible from member functions of class AUTHOR.
iv) How many bytes will be required by an object belonging to class AUTHOR?

DATA FILEHANDLING IN C++

File. The information / data stored under a specific name on a storage device, is called a file.
Stream. It refers to a sequence of bytes.
Text file. It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.
Binary file. It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

Classes for file stream operation

ofstream Stream class to write on files
ifstream : Stream class to read from files
fstream: Stream class to both read and write from/to files.

Opening a file

OPENING FILE USING CONSTRUCTOR
ofstream FOUT(“RESULTS .TXT”); //output only
ifstream FIN(“DA TA.TXT”); //input only

OPENING FILE USING open()
STREAM-OBJECT.open(“FILENAME”, mode)
ofstream OFILE;
OFILE.open(“DATA1.TXT”);

ifstream IF ILE;
IFILE.open(“DATA2.TXT”);

File mode parameter	Meaning
ios::app	Append to end of file
ios::ate	go to end of file on opening
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only

All these flags can be combined using the bitwise operator OR (|). For example, if we want to open the file example.bin in binary mode to add data we could do it by the following call to member function open():

```
fstream FILE;
FILE.open ("EXAMPLE.BIN", ios::out | ios::app | ios::binary);
```

CLOSING FILE

```
FOUT.close();
FIN.close();
```

INPUT AND OUTPUT OPERATION

put() and get() funct ion
the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated str eam.
example :

```
FILE.get(ch);
FILE.put(ch);
```

write() and read() function
write() and read() functions write and read blocks of binary data.
example:

```
FILE.read((char *)&OBJ, sizeof(OBJ));
FILE.write((char *)&OBJ, sizeof(OBJ));
```

FILE POINTERS AND THEIR MANIPULATION

All i/o streams objects have, at least, one internal stream pointer:
ifstream, like istream, has a pointer known as the get pointer that points to the element to be read in the next input operation.
ofstream, like ostream, has a pointer known as the put pointer that points to the location where the next element has to be written.
Finally, fstream, inherits both, the get and the put pointers, from iostream (which is itself derived from both istream and ostream).

These internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

seekg()	moves get pointer(input) to a specified location
seekp()	moves put pointer (output) to a specified location
tellg()	gives the current position of the get pointer
tellp()	gives the current position of the put pointer

The other prototype for these functions is:

```
seekg(offset, refposition );
seekp(offset, refposition );
```

The parameter offset represents the number of bytes the file pointer is to be moved from the location specified by the parameter refposition. The refposition takes one of the following three constants defined in the ios class.

ios::beg start of the file
ios::cur current position of the pointer
ios::end end of the file
example:
FILE.seekg(-10, ios::cur);

BASIC OPERATION ON TEXT FILE IN C++

Program to write in a text file

```
#include<fstream.h>
int main()
{
ofstream fout;
fout.open("out.txt");
char str[300]="Time is a great teacher but unfortunately it kills all its pupils. Berlioz";
fout<<str;
fout.close();
return 0;
}
```

Program to read from text file and display it

```
#include<fstream.h>
#include<conio.h>
int main()
{
ifstream fin;
fin.open("out.txt");
char ch;
while(!fin.eof())
{
fin.get(ch);
cout<<ch;
}
fin.close();
getch();
return 0;
}
```

Program to count number of characters.

```
#include<fstream.h>
#include<conio.h>
int main()
{
ifstream fin;
fin.open("out.txt");
clrscr();
char ch; int count=0;
while(!fin.eof())
{
fin.get(ch);
count++;
}
cout<<"Number of characters in file is "<<count;
fin.close();
getch();
return 0;
}
```

Program to count number of words

```
#include<fstream.h>
#include<conio.h>
int main()
{
ifstream fin;
fin.open("out.txt");
char word[30]; int count=0;
while(!fin.eof())
{
fin>>word;
count++;
}
cout<<"Number of words in file is "<<count;
fin.close();
getch();
return 0;
}
```

Program to count number of lines

```
#include<fstream.h>
#include<conio.h>
int main()
```

```
{
ifstream fin;
fin.open("out.txt");
char str[80]; int count=0;
while(!fin.eof())
{
fin.getline(str,80);
count++;
}
cout<<"Number of lines in file is "<<count;
fin.close();
getch();
return 0;
}
```

Program to copy cont ents of f ile to another file.

```
#include<fstream.h>
int main()
{
ifstream fin;
fin.open("out.txt");
ofstream fout;
fout.open("sample.txt");
char ch;
while(!fin.eof())
{
fin.get(ch);
fout<<ch;
}
fin.close();
return 0;
}
```

AS SI GNME NT TE XT F ILE

QUESTION 5. Write a function to count number of words in a text file named "OUT.TXT"

QUESTION 6. Write a function in C++ to print the count of word the as an independent word in a text file STORY.TXT.
for example, if the content of the file STORY.TXT is
There was a monkey in the zoo. The monkey was very naughty.

Then the output of the program should be 2.

QUESTION 7. Write a function in C++ to count and display the number of lines not starting with alphabet 'A' present in a text file "STORY.TXT".
Example:
If the file "STORY.TXT" contains the following lines,
The rose is red.
A girl is playing there.
There is a playground.
An aeroplane is in the sky.
Numbers are not allowed in the password.

The function should display the output as 3.

BAS I C OP ER ATIO N O N B INARY F ILE I N C ++

```
class student
{
int admno;
char name[20];
public:
void getdata()
{
cout<<"\nEnter The admission no. ";
cin>>admno;
cout<<"\n\nEnter The Name of The Student ";
gets(name);
}
void showdata()
{
cout<<"\nAdmission no. : "<<admno;
cout<<"\nStudent Name : ";
puts(name);
}
int retadmno()
{
return admno;
}
};
```

function to write in a binary f ile

```
void write_data()
{
student obj;
ofstream fp2;
fp2.open("student.dat",ios::binary|ios::app);
obj.getdata();
fp2.write((char*)&obj,sizeof(obj));
fp2.close();
}
```

function to display records of f ile

```
void display()
{
student obj;
ifstream fp1;
fp1.open("student.dat",ios::binary);
while(fp1.read((char*)&obj,sizeof(obj)))
{
obj.showdata();
}
}
fp.close();
}
```

Function to search and display from binary file

```
void search (int n)
{
student obj;
ifstream fp1;
fp1.open("student.dat",ios::binary);
while(fp1.read((char*)&obj,sizeof(obj)))
{
if(obj.retadmno()==n)
obj.showdata();
}
fp1.close();
}
```

QUESTION 8.

Given a binary file STUDENT.DAT, containing records of the following class Student type

```
class Student
{
char S_Admno[lO];      //Admission number of student
char S_Name[30];          //Name of student
int Percentage;                  //Marks Percentage of student
public:
void EnterData()
{
gets(S_Admno);
gets(S_Name);
cin>>Percentage;
}
void DisplayData()
{
cout<<setw(12)<<S_Admno;
cout<<setw(32)<<S_Name;
cout<<setw(3)<<Percentage<<endl;
}
int ReturnPercentage()
{return Percentage;}
};
```

a) Write a function in C++ to add more record in file STUDENT.DAT.

b) Write a function in C++, that would read contents of file STUDENT.DAT and display the details of those Students whose Percentage is above 75.

ARRAY

An array is a collection of data elements of same data type. It is described by a single name and each element of an array is refer enced by using array name and its subscript no.

Declaration of Array

```
Type arrayName[numberOfElements];
```

For example,

```
int Age[5] ;
```

Initialization of One Dimensional Array

An array can be initialized along with declar ation. For array initialization it is required to place the elements separated by commas enclosed within braces.

```
int A[5] = {11,2,23,4,15};
```

It is possible to leave the array size open. The compiler will count the array size.

```
int B[] = {6,7,8,9,15,12};
```

Referring to Array Elements

In any point of a program in which an array is visible, we can access the value of any of its elements individually as if it was a normal variable, thus being able to both read and modify its value. The format is as simple as:

```
name[index]
```

Examples:

```
cout<<age[4];     //print an array element
age[4]=55;              // assign value to an array element
cin>>age[4];          //input element 4
```

Using Loop to input an Array from user

```
int age [10] ;
for (i=0 ; i<10; i++)
{
cin>>age[i];
}
```

Arrays as Parameters

At some moment we may need to pass an array to a function as a parameter. In C++ it is not possible to pass a complete block of memory by value as a parameter to a function, but we are allowed to pass its address.

For example, the following function:

```
void print(int A[])
```

accepts a parameter of type "array of int" called A.
In order to pass to this function an array declared as:

```
int arr[20];
```

we need to write a call like this:

```
print(arr);
```

Here is a complete example:

```
#include <iostream.h>
void print(int A[], int length)
{
for (int n=0; n<length; n++)
cout << A[n] << " ";
cout << "\n";
}
void main ()
{
int arr[] = {5, 10, 15};
print(arr,3);
}
```

BAS IC O PE RATI ON ON O NE D I ME NS IO NAL ARRAY

Function to traverse the array A

```
void display(int A[], int n)
{
cout<<"The elements of the array are:\n";
for(int i=0;i<n;i++)
cout<<A[i];
}
```

Function to Read elements of the array A

```
void Input(int A[], int n)
{
cout<<"Enter the elements:";
for(int i=0;i<n;i++)
cin>>A[i];
}
```

Function to Search for an element from A by Linear Search

```
int Lsearch(int A[], int n, int Data)
{
int I;
for(I=0; I<n; I++)
{
if(A[I]==Data)
{
cout<<"Data Found at : "<<I;
return;
}
}
cout<<"Data Not Found in the array"<<endl;
}
```

Function to Search for an element from Array A by Binary Search

```
int BsearchAsc(int A[], int n, int data)
{
int Mid,Lbound=0,Ubound=n-1,Found=0;
while((Lbound<=Ubound) && !(Found))
{
Mid=(Lbound+Ubound)/2;          //Searching The Item
if(data>A[Mid])
Lbound=Mid+1;
else if(data<A[Mid])
Ubound=Mid-1;
else
Found++;
}
if(Found)
return(Mid+1);          //returning 1ocation, if present
else
return(-1);          //returning -1,if not present
}
```

Function to Sort the array A by Bubble Sort

```
void BSort(int A[], int n)
{
int I,J,Temp;
for(I=0;I<n-1;I++) //sorting
{
for(J=0;J<(n-1-I);J++)
if(A[J]>A[J+1])
{
Temp=A[J]; //swapping
A[J]=A[J+1];
A[J+1]=Temp;
}
}
}
```

Function to Sort the array ARR by Insertion Sort

```
void ISort(int A[], int n)
{
int I,J,Temp;
for(I=1;I<n;I++) //sorting
{
Temp=A[ I];
J=I-1;
while((Temp<A[J]) && (J>=0))
{
A[J+1]=A[J];
J--;
}
A[J+1]=Temp;
}
}
```

Function to Sort the array ARR by Selection Sort

```
void SSort(int A[], int n)
{
int I,J,Temp,Small;
for(I=0;I<n-1;I++)
{
Small=I;
for(J=I+1;J<n;J++) //finding the smallest element
if(A[J]<A[Small])
Small=J;
if(Small!=I)
{
Temp=A[ I]; //Swapping
A[I]=A[Small];
A[Small]=Temp;
}
}
}
```

TWO DIMENSIONAL ARRAYS (C++ implementation)

Function to read the array A

```
void Read(int A[][20], int N, int M)
{
for(int R=0;R<N;R++)
for(int C=0;C<M;C++)
{
cout<<"(R<<','<<")?";
cin>>A[R][C];
}
}
```

Function to display content of a two dimensional array A

```
void Display(int A[][20],int N, int M)
{
for(int R=0;R<N;R++)
{
for(int C=0;C<M;C++)
cout<<A[R][C];
cout<<endl;
}
}
```

Function to find the sum of two dimensional arrays A and B

```
void Addition(int A[][20], int B[][20],int N, int M)
{
for(int R=0;R<N;R++)
for(int C=0;C<M;C++)
C[R][C]=A[R][C] +B[R][C];
}
```

Function to multiply two dimensional arrays A and B of order NxL and LxM

```
void Multiply(int A[][20], int B[][20], int C[][20],int N, int L, int M)
{
for(int R=0;R<N;R++)
for(int C=0;C<M;C++)
{
C[R][C]=0;
for(int T=0;T<L;T++)
C[R][C]+=A[R][T]*B[T][C];
}
}
```

Function to find & display sum of rows & sum of cols. of a 2 dim. array A

```
void SumRowCol(int A[][20], int N, int M)
{
for(int R=0;R<N;R++)
{
int SumR=0;
for(int C=0;C<M;C++)
SumR+=A[R][C];
cout<<"Row("<<R<<") ="<<SumR<<endl;
}
for(int R=0;R<N;R++)
{
int SumR=0;
for(int C=0;C<M;C++)
SumR+=A[R][C];
cout<<"Row("<<R<<")="<<SumR<<endl;
}
}
```

Function to find sum of diagonal elements of a square matrix A

```
void Diagonal(int A[][20], int N, int &Rdiag, int &LDiag)
{
for(int I=0,Rdiag=0;I<N;I++)
Rdiag+=A[I][I];
for(int I=0,Ldiag=0;I<N;I++)
Ldiag+=A[N-I-1][I];
}
```

Function to find out transpose of a two dimensional array A

```
void Transpose(int A[][20], int B[][20],int N, int M)
{
for(int R=0;R<N;R++)
for(int C=0;C<M;C++)
B[R][C]=A[C][R];
}
```

Address Calculation of 2D array

ROW MAJOR

LOC (A[I][J]) = BASE(A) + W*[C*(I- LBI) + (J-LBJ)]

LOC (A[I][J]) = BASE(A) + W*[C*I + J]

COLUMN MAJOR

LOC (A[I][J]) = BASE(A) + W*[R*(J-LBJ) + (I-LBI)]

LOC (A[I][J]) = BASE(A) + W*[R*J + I]

AS SI GNM ENT

QUESTION 9. Write a function in C++ which accepts an integer array and its size as arguments/parameters and exchanges the values of first half side elements with the second half side elements of the array.
Example:
If an array of eight elements has initial content as
2,4,1,6,7,9,23,10
The function should rearrange the array as
7,9,23,10,2,4,1,6

QUESTION 10. Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column.
[Assuming the 2D Array to be a square matrix with odd dimension i.e. 3x3, 5x5, 7x7 etc...]
Example, if the array contents is

```
3   5   4
7   6   9
2   1   8
```

Output through the function should be:
Middle Row : 7 6 9
Middle column : 5 6 1

QUESTION 11. An array P[20][30] is stored in the memory **along the column** with each of the element occupying 4 bytes, find out the memory location for the element P[5][15], if an element P[2][20] is stored at the memory location 5000.

QUESTION 12. An array P[20][30] is stored in the memory **along the row** with each of the element occupying 4 bytes, find out the memory location for the element P[5][15], if an element P[2][20] is stored at the memory location 5000.

POINTER

C++ Memory Map

Once a program is compiled, C++ creates four logically distinct regions of memory:
Code Area : Area to hold the compiled program code
Data Area : Area to hold global variables
Stack Area : Area to hold the return address of function calls, argument passed to the functions, local variables for functions and the cur rent state of the CPU.
Heap : Area from which the memory is dynamically allocated to the program.

Accessing address of a variable

Computer s memory is organized as a linear collection of bytes. Every byte in the computer s memory has an address. Each variable in program is stored at a unique address. We can use address operator & to get address of a variable:

```
int num = 23;
cout << &num;          // prints address in hexadecimal
```

POINTER

A pointer is a variable that holds a memory address, usually the location of another variable in memory.

Defining a Pointer Variable

int *iptr;
iptr can hold the address of an int

Pointer Variables Assignment:

```
int num = 25;
int *iptr;
iptr = &num;
```

To access num using iptr and indirection operator *

```
cout << iptr;          // prints 0x4a00
cout << *itptr;      // prints 25
```

Similary, following declaration shows:

```
char *cptr;
float *fptr;
```

cptr is a pointer to character and fptr is a pointer to float value.

Pointer Arithmetic

Some arithmetic operators can be used with pointers:
- Increment and decrement operators ++, --

- Integers can be added to or subtracted from
pointers using the operators +, -, +=, and -=

Each time a pointer is incremented by 1, it points to the memory location of the next element of its base type.
If “p” is a character pointer then “p++” will increment “p” by 1 byte.
If “p” were an integer pointer its value on “p++” would be incremented by 2 bytes.

Pointers and Arrays

Array name is base address of arr ay
int vals[] = {4, 7, 11};
cout << vals; // displays 0x4a00
cout << vals[0] ; // displays 4

Lets takes an example:

int arr[]={4,7,11};
int *ptr = arr;
What is ptr + 1?
It means (address in ptr) + (1 * size of an int)
cout << *(ptr+1); // displays 7
cout << *(ptr+2); // displays 11

Array Access
Array notation arr[i] is equivalent to the pointer notation *(arr + i)

Assume the variable definitions
int arr[]={4,7,11};
int *ptr = arr;
Examples of use of ++ and --
ptr++; // points at 7
ptr--; // now points at 4

Character Pointers and Strings

Initialize to a character string.
char* a = “Hello”;
a is pointer to the memory location where „H is stored. Here “a” can be viewed as a character array of size 6, the only difference being that a can be reassigned another memory location.
char* a = “Hello”;

a	gives address of „H
*a	gives „H
a[0]	gives „H
a++	gives address of „e
*a++	gives „e

Pointers as Function Parameters

A pointer can be a parameter. It works like a reference parameter to allow change to argument from within function

Pointers as Function Parameters

```
void swap(int *x, int *y)
{
int temp;
temp = *x;
*x = *y;
*y = temp;
}
swap(&num1, &num2);
```

Pointers to Constants and Constant Pointers

Pointer to a constant : cannot change the value that is pointed at
Constant pointer : address in pointer cannot change once pointer is initialized

Pointers to Structures

We can create pointers to structure variables

```
struct Student {int rollno; float fees;};
Student stu1;
Student *stuPtr = &stu1;
(*stuPtr).rollno= 104;
```

-or-

Use the form ptr->member:

```
stuPtr->rollno = 104;
```

Static allocation of memory

In the static memory allocation, the amount of memory to be allocated is predicted and preknown. This memory is allocated during the compilation itself. All the declared variables declared normally, are allocated memory statically.

Dynamic allocation of memory

In the dynamic memory allocation, the amount of memory to be allocated is not known. This memory is allocated during run-time as and when required. The memory is dynamically allocated using new operator.

Free store

Free store is a pool of unallocated heap memory given to a program that is used by the program for dynamic allocation during execution.

Dynamic Memory Allocation

We can allocate stor age for a variable while program is running by using new operator

To allocate memory of type integer
int *iptr=new int;

To allocate array
double *dptr = new double[25] ;

To allocate dynamic structure variables or objects
Student sptr = new Student;   //Student is tag name of structure

Releasing Dynamic Memory

Use delete to free dynamic memory
delete iptr;
To free dynamic array memory
delete [] dptr;
To free dynamic structure
delete Student;

Memory Leak

If the objects, that are allocated memor y dynamically, are not deleted using delete, the memory block remains occupied even at the end of the program. Such memor y blocks are known as orphaned memory blocks. These orphaned memory blocks when increase in number, bring adverse effect on the system. This situation is called memory leak

Self Referential Structure

The self referential structures are structures that include an element that is a pointer to another structure of the same type.

```
struct node
{
int data;
node* next;
}
```

AS SI GNM ENT

QUESTION 13. What will be the output of the following program :

```
#include<iostream.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
```

```
void ChangeString(char Text[], int &Counter)
{
char *Ptr = Text;
int Length = strlen (Text);
for ( ;Counter<Length-2; Counter+=2, Ptr++)
{
* (Ptr + Counter) = toupper( * (Ptr + Counter) );
}
}
void main()
{
clrscr();
int Position = 0;
char Message[] = “Pointers Fun”;
ChangeString (Message, Position);
cout<<Message<<“ @ “<<Position;
}
```

DATA STRUCTURE

STATI C   STAC K

```
#include<iostream.h>
#include<conio.h>
#define size 4

class stack
{
int data[size];
int top;
public:
stack()
{
top=-1;
}
void push();
void pop();
void display();
};
```

```
void stack::push()
{
if(top==size-1)
{
cout<<"\nStack is full";
return;
}
else
{
top++;
cout<<"Enter Data : ";
cin>>data[top];
}
}

void stack::pop()
{
if(top==-1)
cout<<"\n Stack is empty";
else
{
cout<<data[top]<<"deleted "<<endl;
top--;
}
}

void stack::display()
{
int t=top;
while(t>=0)
{
cout<<data[t]<<endl;
t--;
}
}

void main()
{
stack st;
int ch;
do
{
cout<<"\n1. Push\n2. Pop\n3. Display \n4.Quit\nEnter Choice(1-4) ";
cin>>ch;
switch(ch)
{
```

```
case 1: st.push();break;
case 2: st.pop();break;
case 3: st.display();
}
}while(ch!=4);
}
```

STATI C CI RCU LAR Q UE U E

```
#include<iostream.h>
#include<conio.h>
#define size 4

class cqueue
{
int data[size];
int front,rear;
public:
cqueue()
{
front=-1;rear=-1;
}
void insert();
void remove();
};
```

```
void cqueue::insert()
{
if(rear==size-1&&front==0 || front==rear+1)
{
cout<<"\nCircular queue is full";
return;
}
else if(rear==-1)
{
rear++;
front++;
}
else if(rear==size-1)
rear=0;
else
rear++;

cout<<"Enter Data : ";
cin>>data[rear];
}
```

```
void cqueue::remove()
{
if(front==-1)
{
cout<<"\n Circular Queue is empty";return;
}

cout<<data[front] <<" deleted"<<endl;

if(front==rear)
{
front=-1;rear=-1;
}
else if(front==size-1)
front=0;
else
front++;
}

void main()
{
cqueue cq;
int ch;
do
{
cout<<"\n1. Insert\n2. Remove\n3. Quit\nEnter Choice(1-3) ";
cin>>ch;
switch(ch)
{
case 1: cq.insert();break;
case 2: cq.remove();break;
}
}while(ch!=3);
}
```

D YNAM IC S TACK

Stack is a linear data structure in which insertion and deletion of elements takes place only one end known as TOP.

```
#include<iostream.h>
#include<conio.h>
```

```
struct node
{
int data;
node *next;
};

class stack
{
node *top;
public :
stack()
{ top=NULL;}
void push();
void pop();
void display();
~stack();
};

void stack::push()
{
node *temp;
temp=new node;
cout<<"Enter data :";
cin>>temp->data;
temp->next=top;
top=temp;
}

void stack::pop()
{
if(top!=NULL)
{
node *temp=top;
top=top->next;
cout<<temp->data<<"deleted";
delete temp;
}
else
cout<<"Stack empty";
}

void stack::display()
{
node *temp=top;
while(temp!=NULL)
{
```

```
cout<<temp->data<<" ";
temp=temp->next;
}
}

stack::~stack()
{
while(top!=NULL)
{
node *temp=top;
top=top->next;
delete temp;
}
}

void main()
{
stack st;
char ch;
do
{
cout<<"stack options\nP for push \nO for Pop \nD for Display \nQ for quit";
cin>>ch;
switch(ch)
{
case 'P': st.push();break;
case 'O': st.pop();break;
case 'D': st.display();break;
}
}while(ch!='Q');
}
```

DY NAM IC Q UE UE

Queue is a linear data structure in which insertion and deletion of elements takes place from two opposite ends rear and front respectively.

```
#include<iostream.h>
#include<conio.h>

struct node
{
int data;
node *next;
};
```

```
class queue
{
node *rear,*front;
public:
queue()
{ rear=NULL;front=NULL;}
void qinsert();
void qdelete();
void qdisplay();
~queue();
};

void queue::qinsert()
{
node *temp;
temp=new node;
cout<<"Data :";
cin>>temp->data;
temp->next=NULL;
if(rear==NULL)
{
rear=temp;
front=temp;
}
else
{
rear->next=temp;
rear=temp;
}
}

void queue::qdelete()
{
if(front!=NULL)
{
node *temp=front;
cout<<front->data<<"deleted \n";
front=front->next;
delete temp;
if(front==NULL)
rear=NULL;
}
else
cout<<"Queue Empty..";
}
```

```
void queue::qdisplay()
{
node *temp=front;
while(temp!=NULL)
{
cout<<temp->data<<endl;
temp=temp->next;
}
}

queue::~queue()
{
while(front!=NULL)
{
node *temp=front;
front=front->next;
delete temp;
}
}

void main()
{
queue obj; char ch;
do
{
cout<< "i. insert\nd. Delete\ns. Display\n q. quit ";
cin>>ch;
switch(ch)
{
case 'i' : obj.qinsert();break;
case 'd' : obj.qdelete();br eak;
case 's' : obj.qdisplay();
}
}while(ch!='q');
}
```

AS SI GNM ENT

QUESTION 14. Change the following infix expression postfix expression.
(A + B)*C+D/E- F

QUESTION 15. Evaluate the following postfix expression using a stack and show the contents of stack after execution of each oper ation :
50,40,+,18, 14,-, *

C++ BASIC CONCEPTS (REVISION TOUR OF CLASS 11)

Question: Why main function is special in C++ ?
Answer: Whenever a C++ program is executed, execution of the program starts and ends at main(). The main is the driver function of the program. If it is not present in a program, no execution can take place.

Question: What is run-time error, logical error and syntax error?
Answer: Syntax error - the errors which are traced by the compiler during compilation, due to wrong grammar for the language used in the program, are called syntax errors.
For example, cin>>a; // instead of extraction operator insertion operator is used.
Run time Error - The errors encounter ed during execution of the program, due to unexpected input or output are called run-time error.
For example - a=n/0; // division by zero
Logical Error - These errors are encountered when the program does not give the desired output, due to wrong logic of the program.
For example : remainder = a+b // instead of using % operator + operator is used.

Question: What is the role of #include directive in C++?
Answer: The preprocessor directive #include tells the complier to insert another file into your source file. In effect, #include directive is replaced by the contents of the file indicated.

Question: What is compiler and linker?
Answer: Compiler - It is a program which converts the program written in a programming language to a program in machine language.
Linker - It is a program which links a complied program to the necessary library routines, to make it an executable program.

Question: Why is char often treated as integer data type in C++ ?
Answer: The memory implementation of char data type is in terms of the number code. Therefore, it is said to be another integer data type.

Question: What is type conversation in C++ ?
Answer: When two operands of different data types are encountered in the same expression, the variable of lower data type is automatically converted to the data tpes of variable with higher data type, and then the expression is calculated.
For example: int a=98; float b=5; cout<<a/3.0; //converts to float type, since 3.0 is of float type.
cout<<a/b; // converts a temporarily to float type, since b is of float type, and gives the result 19.6.

Question: What is type casting in C++ ?
Answer: Type casting refers to the data type conversions specified by the programmer, as opposed to the automatic type conversions. This can be done when the compiler does not do the conversions automatically. Type casting can be done to higher or lower data type.
For example : cout<<(float)12/5; //displays 2.4, since 12 is converted to float type.

Question: What is the effect of absence of break in switch case statement in C++ ?
Answer: The break keyword causes the entire switch statement to exit, and the control is passed to statement following the switch.. case construct. Without break, the control passes to the statements for the next case. The break statement is optional in switch..case construct.

Question: In a control structure switch-case what is the purpose of default in C++ ?
Answer: This keyword gives the switch…case construct a way to take an action if the value of the switch variable does not match with any of the case constants. No break statement is necessary after default case, since the control is already at the end of switch..case construct. The default is optional in case of switch…case construct.

Question: What is the difference between while and do-while loop in C++ ?
Answer: While is an Entry Controlled Loop, the body of the loop may not execute even once if the test expression evaluates to be false the first time, whereas in do..while, the loop is executed at least once whether the condition holds true the first time or not.

Question: What is the differ ence between call by value and call by reference in a user defined function in C++?
Answer: The value of the actual parameters in the calling function do not get affected when the arguments are passed using call by value method, since actual and formal parameters have different memory locations.
The values of the formal parameters affect the values of actual parameters in the calling function, when the arguments are passed using call by reference method. This happens since the formal parameters are not allocated any memory, but they refer to the memory locations of their corresponding actual parameters

Question: What is preprocessor directive?
Answer: A preprocessor directive is an instruction to the complier itself. A part of compiler called preprocessor deals with these directives, before real compilation process. # is used as preprocessor directive in C++.

Question: What is the differ ence between local variable and global variable?
Answer: Local variables are those variables which are declared within a function or a compound statement and these variables can only be used within that function/scope. They cannot be accessed from outside the function or a scope of it s declaration. This means that we can have variables with the same names in dif ferent functions/scope. Local variables are local to the function/scope in which they are declared.
Global variables are those variables which are declared in the beginning of the program. They are not declared within a function. So, these variables can be accessed by any function of the program. So, global variables are global to all the functions of the program.

Question: What is the role of #define in C++?
Answer: It is a preprocessor directive to define a macro in a C++ program. Macros provide a mechanism for token replacement with or without a set of formal, function line parameters. For example :
#define PIE 3.1416
#define AVG(A,B,C) (A+B+C)/3

Question: What are the major differences between Object Oriented Programming and Procedural Programming?

Object Oriented Programming	Procedural Programming
Emphasis on data	Emphasis on doing things (function)
Follow bottom up approach in program design	Follow top-down approach in program design
Concept of Data hiding prevents accidental change in the data	Due to presence of global variables, there is possibilities of accidental change in data.
Polymorphism, inheritance, Data Encapsulation possible	Not applicable

LIBRARY FUNCTION

# Include Directive
The # include directive instructs the compiler to read and include another file in the current file. The compiler compiles the entire code. A header file may be included in one of two ways.

include <iostream.h>
or
include "iostream.h"

The header file in angle brackets means that file reside in standard include directory. The header file in double quotes means that file reside in current directory.

LIBRARY FUNCTION
C++ provides many built in functions that saves the programming time

Mathematical Functions
Some of the important mathematical functions in header file **math.h** are

Function	**Meaning**
sin(x)	Sine of an angle x (measured in radians)
cos(x)	Cosine of an angle x (measured in radians)
tan(x)	Tangent of an angle x (measured in radians)
asin(x)	Sin-1 (x) where x (measured in radians)
acos(x)	Cos-1 (x) where x (measured in radians)

exp(x)	Exponential function of x (ex)
log(x)	logarithm of x
log 10(x)	Logarithm of number x to the base 10
sqrt(x)	Square root of x
pow(x, y)	x raised to the power y
abs(x)	Absolute value of integer number x
fabs(x)	Absolute value of real number x

Character Functions
All the character functions require **ctype.h** header file. The following table lists the function.

Function	**Meaning**
isalpha(c)	It returns True if C is an uppercase letter and False if c is lowercase.
isdigit(c)	It returns True if c is a digit (0 through 9) otherwise False.
isalnum(c)	It returns True if c is a digit from 0 through 9 or an alphabetic character (either uppercase or lowercase) otherwise False.
islower(c)	It returns True if C is a lowercase letter otherwise False.
isupper(c)	It returns True if C is an uppercase letter otherwise False.
toupper(c)	It converts c to uppercase letter.
tolower(c)	It converts c to lowercase letter.

String Functions
The string functions are present in the **string.h** header file. Some string functions are given below:

strlen(S)	It gives the no. of characters including spaces present in a string S.
strcat(S1, S2)	It concatenates the string S2 onto the end of the string S1. The string S1 must have enough locations to hold S2.
strcpy(S1, S2)	It copies character string S2 to string S1. The S1 must have enough storage locations to

	hold S2.

strcmp((S1, S2)==0)
strcmp((S1, S2)>0)
strcmp((S1, S2) <0)
	It compares S1 and S2 and finds out whether S1 equal to S2, S1 greater than S2 or S1 less than S2.

strcmpi((S1, S2)==0)
strcmpi((S1, S2)>0)
strcmpi((S1, S2) <0)
	It compares S1 and S2 ignoring case and finds out whether S1 equal to S2, S1 greater than S2 or S1 less than S2.

strrev(s) It converts a string s into its reverse

strupr(s) It converts a string s into upper case

strlwr(s) It converts a string s into lower case

Console I/O functions

The following are the list of functions are in **stdio.h**

getchar() It r eturns a single character from a standard input device (keyboard). It takes no parameter and the returned value is the input character.

putchar() It takes one argument, which is the character to be sent to output device. It also returns this character as a result.

gets() It gets a string terminated by a newline character from the standard input stream stdin.

puts() It takes a string which is to be sent to output device.

General purpose standard library functions

The following are the list of functions are in **stdlib.h**

randomize() It initializes / seeds the random number generator with a random number

random(n) It generates a random number between o to n-1

atoi(s) It converts string s into a numerical representation.

itoa(n) It converts a number to a string

Some More Functions

The getch() and getche() functions
The general for of the getch() and getche() is

```
ch=getche();
ch1=getch();
```

ch and ch1 are the variables of type character. They take no argument and require the **conio.h** header file. On execution, the cursor blinks, the user must type a character and press enter key. The value of the character returned from getche() is assigned to ch. The getche() fuction echoes the character to the screen. Another function, getch(), is similar to getche() but does not echo character to the screen.

AS SI GNM ENT

QUESTION 16.

Observe the following program SCORE.CPP carefully, if the value of Num entered by the user is 5, choose the correct possible output(s) from the options from (i) to (iv), and justify your option. 2

```
//program : SCORE.CPP
#include<stdlib.h>
#include<iostream.h>
void main()
{
randomize ();
int Num, Rndnum;
cin>>Num;
Rndnum = random (Num) + 5;
for (int N = 1; N<=Rndnum; N++)
cout<<N<<" ";
}
```

Output Options:
(i) 1 2 3 4
(ii) 1 2
(iii) 1 2 3 4 5 6 7 8 9
(iv) 1 2 3

QUESTION 17. Give the output of the following program segment

```
char *NAME = "CoMPutER";
for (int x = 0: x < strlen(NAME); x++)
if (islower(NAME [x]))
NAME [x] = toupper(NAME[x]);
else
if (isupper(NAME[x]))
if (x%2 ==0)
NAME[x] = tolower(NAME[x]);
else
NAME[x] =NAME[x-1];
puts(NAME);
```

SOLUTION OF ASSIGNMENTS

Solution of Question 1.

```
Col50Row50
Col50Row70
Col25Row50
```

Solution of Question 2.

```
class student
{
int admno;
char sname[20];
float eng, math, science, total;
float ctotal()
{
return eng+math+science;
}
public :
void Takedata()
{
cout<<"Enter admission number ";
cin>>admno;
cout<<"Enter student name";
gets(sname);
cout<<"Enter marks in english, maths and science ";
cin>>eng>>math>>science;
total=ctotal();
}
void Showdata()
{
cout<<"\nAdmission number "<<admno<<"\nStudent Name "<<sname<<"\nMarks
in english "<<eng<<"\nMath "<<math<<"\nScience "<<science<<"\nTotal "<<total;
}
};
```

Solution of Question 3.

i) Exam obj(98);
Exam obj("Maths",89);

ii) Function overloading or polymorphism

iii) Destructor. It invokes automatically when object goes out of its scope.

iv) Copy constructor.

```
Exam::Exam(Exam &T)
{
Marks = T.Marks;
strcpy(Subject,T.subject);
}
```

Solution of Question 4.

i) None of the data member is accessible from object of class AUTHOR

ii) Enter(), Display(), Start(), Show()

iii) void Register(), void Enter(), void Display(), float Employees, void Haveit(), void Giveit(), int Acode, char Aname[20], float Amount, void Start(), void Show()

iv) 70

Solution of Question 5.

```
void countword()
{
int count=0;
char word[30];
ifstream fin;
fin.open("OUT.TXT");
while(!fin.eof())
{
fin>>word;
count++;
}
cout<<"The number of words in files are "<<count;
fin.close();
}
```

Solution of Question 6.

```
void countthe()
{
int count=0;
char word[30];
ifstream fin;
fin.open("STORY.TXT");
while(!fin.eof())
{
fin>>word;
if(strcmpi(word,"the")==0)
count++;
}
cout<<"The number of word in files are "<<count;
fin.close();
}
```

Solution of Question 7.

```
void countline()
{
int count=0;
char str[80];
ifstream fin;
fin.open("STORY.TXT");
while(!fin.eof())
{
fin.getline(str,80);
if(str[0]=='A')
count++;
}
cout<<"The number of lines starting with A are "<<count;
fin.close();
}
```

Solution of Question 8.

```
void addrecord()
{
ofstream ofile;
ofile.open("STUDENT.DAT",ios::binary|ios::app);
Student obj;
obj.EnterData();
ofile.write((char*)&obj,sizeof(obj));
ofile.close();
}

void displayrecord()
{
ifstream ifile;
ifile.open("STUDENT.DAT",ios::binary);
Student obj;
while(ifile.read((char*)&obj,sizeof(obj)))
{
if(obj.ReturnPercentage()>75)
obj.DisplayData();
}
ifile.close();
}
```

Solution of Question 9.

```
void exchange(int arr[],int s)
{
int temp, mid=s/2;
for(int i=0;i<s/2;i++,mid++)
{
temp=arr[i];
arr[i]=arr[mid];
arr[mid]=temp;
}
}
```

Solution of Question 10.

```
void mrowmcol(int ARR[3][3],int s)
{
int mid=s/2;
for(int i=0;i<s;i++)
{
cout<<ARR[mid][i]<<" ";
}
cout<<endl;
for(i=0;i<s;i++)
{
cout<<ARR[i][mid]<<" ";
}
}
```

Solution of Question 11.

COLUMN MAJOR FORMULA

LOC (A[I][J]) = BASE(A) + W*[R*J + I]

Given,
LOC(A[2][20])=5000
W=4
I=2
J=20

R=20
C=30
BASE(A)=?

LOC (A[I][J]) = BASE(A) + W*[R*J + I]
5000=BASE(A) + 4[20*20 + 2]
5000=BASE(A) + 4*402
BASE(A) = 5000-1608
BASE(A)=3392

LOC(A[5][15]=?
W=4
I=5
J=15
R=20
C=30
BASE(A)=3392
LOC (A[I][J])= BASE(A) + W*[R*J + I]
=3392 + 4*[20*15 + 5]
=3392 + 4*305
=3392+1220
=4612

Solution of Question 12.

ROW MAJOR FORMULA

LOC (A[I][J]) = BASE(A) + W*[C*I + J]

Given,
LOC(A[2][20])=5000
W=4
I=2
J=20
R=20
C=30
BASE(A)=?

LOC (A[I][J]) = BASE(A) + W*[C*I + J]
5000=BASE(A) + 4[30*2 + 20]
5000=BASE(A) + 4*80
BASE(A) = 5000-320
BASE(A)=4680

LOC(A[5][15]=?
W=4
I=5
J=15
R=20
C=30
BASE(A)=4680
LOC (A[I][J]) = BASE(A) + W*[C*I + J]
=4680 + 4*[30*5 + 15]
=4680 + 4*165
=4680+660
=5340

Solution of Question 13.

PoiNteRs Fun @ 10

Solution of Question 14.

infix expression (A + B)*C+D/E-F

ITEM SCANNED	**STACK**	**OPREATION**	**EXPRESSION**
(	(	PUSH	
A	(		A
+	(+	PUSH	A
B	(+		AB
)		POP	AB+
*	*	PUSH	AB+
C	*		AB+C
+	+	PUSH,POP	AB+C*
D	+		AB+C*D
/	+/	PUSH	AB+C*D
E	+/		AB+C*DE
-	+-	PUSH,POP	AB+C*DE/
F	+-		AB+C*DE/F
	EMPTY	POP	AB+C*DE/F-+

Postfix Expression AB+C*DE/F-+

Solution of Question 15.

EVALUATION OF 20, 30, +, 50, 40, - ,*

ITEM SCANNED	OPERATION	STACK	
20	PUSH 20	20	
30	PUSH 30	20,30	
+	POP 30, POP 20		20+30=50
	PUSH 50	50	
50	PUSH 50	50,50	
40	PUSH 40	50,50,40	
-	POP 40, POP 50		50-40=10
	PUSH 10	50,10	
*	POP 10, POP 50		50*10=500
	PUSH 500	500	
	POP 500		

Ans : 500

Solution of Question 16. (iii)

Solution of Question 17.

cOmmUTee