Q.1 (a) (Definition- 2 marks Example-1 marks Call- 1 mark)

A constructor is aspecial member function whose task is to initialize the objects of its class.

The constructors that can take argument are called parameterized constructors.

Ex: class integer

```
{
intm,n;
public:
integer(intx,int  y);
            ……..
            ……
};
integer :: integer(int  x,int  y)
{
  m=x;
 n =y;
}
```

Parameterized constructor can be called in two ways

1) Explicitly

```
integer int1=integer(0,100);
```

2) Implicitly

```
integer  int1(0,100);
```

b) Some of features of procedure oriented programming (Any four 1 mark each)

- Emphasis is on doing things.
- Large programs are divided into smaller program known as function.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transforms from one form to another.
- Employs top-down approach in program design.

Q.1 (c) (Definition 1 mark)

i) Copy constructor:

A copy constructor is used to declare & initialize the object from another

object.

ii) Access specifiers:

Access specifier specifies that how to access the class members (private, public, protected).

iii) Static data member:

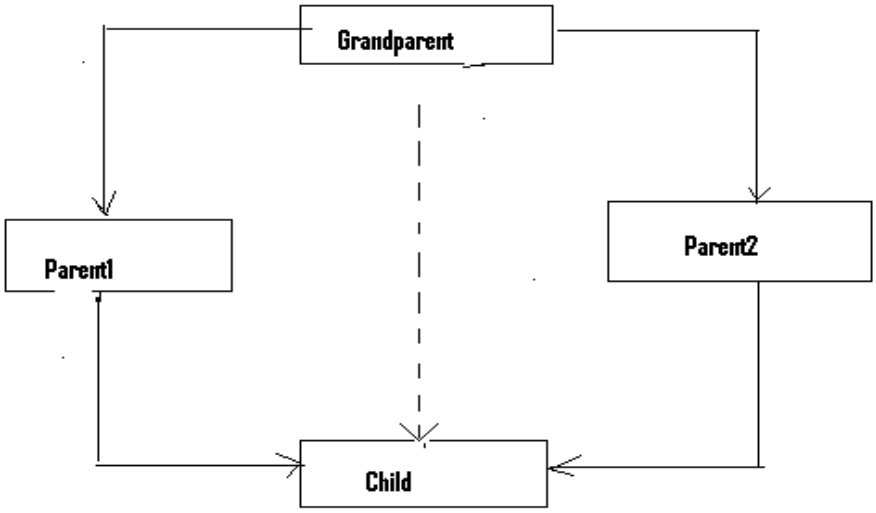
A data member of a class can be qualified as static. Static variable maintain value common to the entire class & its value is initialized to zero when first object is created.

iv) This pointer:

A unique keyword called This to represent an object that invokes a member function.

Q.1 (d) (Diagram-1 mark Explanation- 3 marks)

The duplication of inherited members due to these multiple paths can be avoided by making the common base class (ancestor class) as virtual base class while declaring the direct or intermediate base class.



Multipath Inheritance

Consider a situation where all the three kinds of inheritance, namely, multilevel, multiple & hierarchical inheritance, are involved .this is illustrated in fig above. The child has two direct base class 'parent1' and 'parent2' which themselves have a common base class 'grandparent '. The 'child' inherits traits of 'grandparent' via two separate paths. it can also inherit directly as shown by the broken line. the grandparent is sometimes referred to as indirect base class.

All the public & protected members of 'grandparent' are inherited into child twice first via 'parent1' second via 'parent2'. This means, 'child' would have duplicate sets of the member inherited from 'grandparent'. This introduces ambiguity & should be avoided using virtual base class.

Q.1 (e) (Two methods – 2marks for each method)

A file can be opened in two ways:

1) Using the constructor function of the class:

Here, a filename is used to initialize the file stream object.

Ex: the following statement opens file named "results" for output:

```
ofstreamoutfile("results");          //output only
```

Similarly, the following statement declares infile as an ifstream object & attaches it to the file

Data for reading (input).

```
ifstreaminfile( "Data");             //input only
```

2) Opening files using open()

The function open() can be used to open multiple files that use the same stream object.

File-stream-class stream-object;

Stream-object.open("filename");

Ex: ofsteamoutfile;

outfile.open("data"); //output only

Q.1 (f) Advantages of pointers (1 mark each)

- Can return more than one value from function.

-Increasing the programming performance, using pointers we can access the variables faster.

-In case of arrays, we can decide the size of the array at runtime by allocating the necessary space Or dynamic memory management

-Data structure handling (Linked list, Stack, queue)

Q.1 (g)(Concept-1 mark Program-3 marks)

Same function prototype in base class & derived class then by declaring base classes common function as virtual we can call the both functions by using base pointer this method is known as overriding

```
#include<iostream.h>
class base
        {
public :
void display(){cout<<"\n display base";}
virtual void show(){cout<<"\n show base";}
         };
class    derived :public base
       {Public :
void display()
              {cout<<"\n display derived";}
void  show()
                 {cout<<"\n show derived";}
       };
void main()
        {   base B, *bptr;
derived D;
cout<<"\n bptr points to  base\n";
bptr=&B;
bptr->display();
bptr->show();
cout<<"\n bptr points to derived \n";
bptr=&D;
bptr->display();
bptr->show();
getch();          }
```

Q.2 (a) (Explanation- 2 marks, Example-2 marks)

a) Linking of procedure call to the code is at the time of compilation is known as static binding.

Static binding is achieved by using function overloading. overloading refers to the same thing for diff purposes .this means that we can use the same function name to create functions that can perform variety of tasks.

```
Ex:  #include<iostream.h>

int volume(int s)

{ return (s*s*s);

      }

double  volume( double r,int h)

{ return (3.14*r*r*h);

     }
  void main()

   {

cout<<"volume of cube is"<<volume(5);

cout<<"volume of cylinder is"<<volume(10.5,8);

getch();

   }
```

Q.2 (b) (Correct program carries 4 marks)

```
#include<iostream.h>

#include<conio.h>

class product

 {

private:

                charproduct_name[10] ;

                float  product_price;

public:
```

```
        void accept()
        {
        cout<<"\nentered the product_name,price";
        cin>>product_name>>product_price;
        }
        void display()
        {
        cout<<"\n product_name is:"<<product_name;
        cout<<"\n product_price is:"<<product_price;
        }
    };
void main()
    {
          Product p, *bptr;
        bptr=&p;
bptr->accept();
bptr->display();
getch();
     }
```

Q.2 (c) (Definition 1 mark each)

i) Class is user define that binds data &code to manipulate that data together into single unit.

ii) Objects are the basic runtime entities in object oriented programming.

OR

Objects are the variable of type class.

iii) Array of variables of the type class.

OR

Collection of similar type objects

iv) A member function that declared static can have access to only other static member &they are called using class name instead of objects.

Q.2 (d) (Any four features 1 mark each)

Some of features of object oriented programming

- Emphasis is on data rather than procedure.

-Programs are divided into what are known as objects

-Data structure designed such that they characterize the objects.

-functions that operate on the data of an object are tied together in the data structure.

-Data is hidden & cannot be accessed by external functions.

-Objects may communicate with each others through functions.

-New data and functions can be easily added whenever necessary.

-Follows bottom-up approach in program designing.

Q.2 (e)

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
class binary
{private:
int a;
public:
void accept()
{
cout<<"\nentered the values of  first time  base &  second time power";
cin>>a;
}
binary operator ^ (binary d1)
{binary temp;
temp.a =pow(a,d1.a);
return temp;
```

```
}
void display()
{
cout<<"\npower is:"<<a;
}
};
void main()
{
binary b1,b2,b3;
b1.accept();
b2.accept();
b3=b1^b2;
b3.display();
getch();
 }
```

Q.2 (f) (Types-2 marks, Example- 2 marks)

There are five types of inheritance

1)single inheritance

2)multilevel inheritance

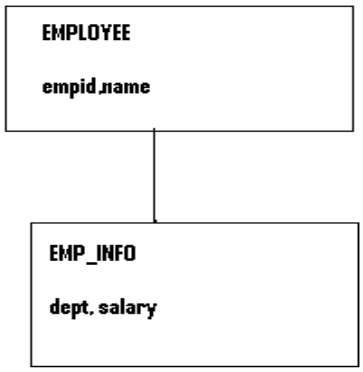
3) multiple inheritance

4)hierarchical inheritance

5)hybrid inheritance

single inheritance

when a derived class with one base class then that type of inheritance is single inheritance




```
#include<iostream.h>
    #include<conio.h>
class EMPLOYEE
     {
private:
                charemp_name[10] ;
                intemp_id;
public:
        void accept()
        {
        cout<<"\nentered the empid,_name";
        cin>>emp_id>>emp_name;
        }
        void display()
        {
        cout<<"\n emp name is:"<<emp_name;
        cout<<"\n emp id  is:"<<emp_id;
        }
   };
```

```
classEMP_info: public EMPLOYEE
    {
private:
                chardept[10] ;
                fioat salary;
public:
        void  getdata()
        {
        cout<<"\nentered the dept name & salary of employee";
        cin>>dept>>salary;
        }
        void show()
        {
        cout<<"\n dept is:"<<dept;
        cout<<"\n salary is:"<<salary;
        }
   };
void main()
{
   EMP_INFO E1;
E1.accept();
E1.getdata();
E1.display();
E1.show();
getch();
}
```

Q.3 (a)

Object oriented programming (OOP) is programming paradigm using "objects". Usually instances of a class consisting data fields and methods together with their interaction to design application and computer program. (1 mark)

Data encapsulation and data abstraction:

Data encapsulation means wrapping of data and functions into class.

- It is not accessible to outside world and only those functions which are wrapped in class can access it.
- Data abstraction refers to act of representing essential features without including background details of explanation.
- Classes use concept of abstraction.
- Classes uses concepts of data abstraction known as Abstract Data Types (ADT) (3 marks).

Q.3 (b) (1 mark for each point)

Sr.No	Call by value	Call by reference
1	Whenever a portion of program invokes a function with formal arguments, control will be transferred from main to calling function and value of actual arguments is copied to function	When a function is called by portion of program the address of actual arguments are copied on to formal arguments, though they may be refereed by different variable name.
2	When control is transferred back from function to calling function of program, the altered values are not transferred back.	Any change i.e. made to data item will be recognized in both function and calling portion of program.
3	In this formal arguments are pass to function	In this arguments are passed through references of address.
4	Suitable example program	Suitable example program

Q.3 (c)(1 mark for each point)

1. When a class containing a virtual function is inherited the derived class redefines the virtual function to fit its own needs.
2. Virtual function implement "one interface multiple methods"
3. It is capable of supporting run-time.
4. When a base pointer to derived object that contains a virtual function, c++ determines which version of that function to call based upon the type of object pointed by the pointer and this is determined at run time.
5. Thus, when different objects are pointed to, different versions of functions are executed. Same effect applies to base class reference.

Q.3 (d)

```
#include<iostream.h>
#include<fstream.h>
int main()
{
ofstreamfout;                    //create output stream
fout.open ("syco.dat");          //connect "syco"
fout<<"OOP\n";
fout<<"AMT\n";
fout<<"RD\n";
fout<<"DT\n";
fout.close();                    //disconnect "syco"
constint N=80;                   //size of line
char line [N]
ifstream fin;                    // create input stream
fin.open("syco.dat")             //connect "syco "
cout<<"Contents of  file\n";
while (fin)                      //check end of file
{
fin.getline(line, N);            // read line
cout<<line;                      // display it
}
fin.close();                     //disconnect "syco"
return 0;
}
```

Output:-

OOP

AMT

RD

DT

OR

```
void main()
{
Ofstreamfout;
fout.open (“syco.dat”);
fout<<”OOP\n”;
fout<<”AMT\n”;
fout<<”RD\n”;
fout<<”DT\n”;
fout.close();
ifstream fin;
fin.open (“syco.dat”);
fin>>OOP;
fin>>AMT;
fin>>RD;
fin>>DT
fin.close() ;
getch() ;
}
```

Q.3 (e)

1. If a static data member is declared as a private category of a class then non-member functions can’t access these members. (1 mark)

2. If a static member is declared as public then any member of class can access. (1 mark)

3. Whenever a static data member is declared and it has only a single copy, it will be shared by all instances of class. (1 mark)

The main advantage of using a static member is to declare global data which should be updated while program lives in the memory. (1 mark)

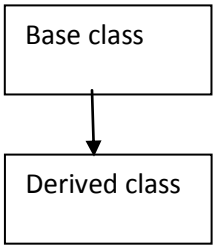
Q.3 (f)

```
#include<iostream.h>
#include<conio.h>
{
private:
int hra, da, basic, sal;
public:
salary(int t, int b, int d=100, int h=200)
{
ta=t;
ba=b;
da=d;
hra=h;
}
void display()
{
sal=ta+da+hra+ba;
cout<<"salary="<<sal;
}
};
void main()
{
salary s(200,300);
s.display();
getch();
}
```

Q.4 (a)

Inheritance is a process by which object of one class can acquire the properties of another class
In c++ class that is inherited is called as base class. The class that does not inherit is called derived class (1 mark)

Ex:-



Ex:- Syntax for derived Class

Class <derived-classname>:<access modifier><base-classname>

“

// body of class

};

Access modifiers are also called as visibility mode:

1. Public
2. Private
3. Protected

1. Public:- When access specifier is public i.e. base class is publicly inherited by derived class all public member of base class become public members of derived class and all protected members become protected in derived class. Proper syntax (1 mark)

2. Private:- All public and protected members of base class become private members of derived class. Proper syntax (1 mark)

3. Protected:- In this all public and protected members of base class become protected of derived class. Proper syntax (1 mark).

Q.4 (b) (4 marks)

```
#include <iostream.h>

#include<conio.h>

void main()

{

char *p, str[15];
```

```
int length=0;
clrscr();
cout<<"Enter string";
cin>>str;
p=str;
while(*p!='\o')
{
length=length+1;
p++;
}
cout<<"length of string is="<<length;
getch();
}
```

Ouput:-

Enter String: syco

Length of string is = 4

Q.4 (c)

- Ability for object of different classes to respond differently to same message (1 mark)
- Two types :- i) Compile-time ii) Run time (1 mark)
- Run time polymorphism is also called as late binding or dynamic binding.
- Sometimes a situation occurs when function name and prototype is same in both base class and in derived class.
- Compiler does not know what to do, which function to call. In this case appropriate member function is selected at run time so known as run time polymorphism or late binding (2 marks)

Q.4 (d)

1. get():- class istream defines a member function get() which is used to handle single character input/output operations at a time. Two types of get() functions are:

get(char *)

get(void) (1 mark)

2. put():- Class ostream defines the members function put() which is used to output a line of text on a screen character by character.

Syntax: cout.put(char) (1 mark)

Ex :- cout.put ('s')

displays character s.

3. getline() :- Function reads a whole line of text that ends with a newline character (1 mark)

Syntax:- cin.getline(line, size)

4. write():- This function is used to write a set of character into file.

Write() is member function of ostream class used to display entire line.

Syntax:- cout.write (line, size)

Line is a variable of which contents has to be display.

Size is no. of characters to be display (1 mark)

Q.4 (e) (Explanation-3 marks, Example-1 mark)

Class is user defined data types which encapsulates data and member function together. For example:

Student s;

- Create a variable of data type student, where student is a class name.
- This variable is called as object.
- We can create any number of object in one statement like, student s1, s2, s3;
- Object is declared after defining a class i.e. after ending brace but before semicolon.

class student

{

}s1,s2,s3;

After creation of object we can access member of class with help of (.) dot operator.

The storage space is allocated for object when object of class is declared.

Q.4(f) (1 mark for each point any four points)

1. Name of destructor is same as class name
2. Before name of destructor tilde sign (~) is attached.
3. This member function in public section only.
4. Does not have return type and does not take any arguments.
5. Definition of destructor does not have any meaningful contents.

Ex:-
Class abc
Will have destructor function like this: ~abc()
{
cout<<:Destructor invoked";
}

6. It can't be declared static, const or volatile.

Q.5 (a)(Logic-4 marks, Syntax-4 marks)

```
#include<iostream.h>
#include<conio.h>
class employee
{
protected:
int emp_id;
char emp_name[20];
};
class fitness:public employee
{
int height;
int weight;
public:
void accept()
 {
cout<<"Enter the employee data:";
cout<<"\nEmployee id:";
cin>>emp_id;
cout<<"\nEmployee name:";
cin>>emp_name;
cout<<"\nEmployee height(in cm):";
```

```
cin>>height;
cout<<"\nEmployee weight:";
cin>>weight;
 }
void display()
 {
cout<<"\n\nEmployee data is:";
cout<<"\nId="<<emp_id;
cout<<"\nName="<<emp_name;
  cout<<"\nHeight="<<height<<" "<<"cm";
cout<<"\nWeight="<<weight;
 }
};
void main()
{
fitness f;
clrscr();
f.accept();
f.display();
getch();
}
```

Q.5 (b) (Logic-4 marks, Syntax-4 marks)

```
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
void main()
{
```

```
const int SIZE=80;
char line[SIZE];
ifstream fin1,fin2;
fin1.open("country");
fin2.open("capital");
for(int i=1;i<=10;i++)
{
if(fin1.eof()!=0)
{
  cout<<"Exit from country\n";
exit(1);
}
fin1.getline(line,SIZE);
cout<<"Capital of "<<line;
if(fin2.eof( )!=0)
{
 cout<<"Exit from capital\n";
exit(1);
}
fin2.getline(line,SIZE);
cout<<line<<"\n";
}
getch();
}
```

Q.5 (c)(Definition-4 marks, Example-4-marks)

1. Private: The members declared as "private" can be accessed only within the same class and not from outside the class.

2. Public: The members declared as "public" are accessible within the class as well as from outside the class.
3. Protected: The members declared as "protected" cannot be accessed from outside the class, but can be accessed from a derived class. This is used when inheritance is applied to the members of a class.

Example1 :-

```
#include<iostream.h>

#include<conio.h>

class product

{

private:

int p_id;

char pname[20];

public:

float price;

void get()

{

 cout<<”Enter product id:”;

cin>>p_id;

cout<<”\nEnter product name”;

cin>>pname;

}

void put()

{

 cout<<”\nProduct data is:”;

cout<<”\nId=”<<p_id;

cout<<”\nName=”<<p_name;

cout<<”\nPrice=”<<price;
```

```
}
};
void main()
{
 product p;
clrscr( );
p.get( );
p.put( );
getch( );
}
```

In above example private members such as p_id and pname are only accessible within a class but they are not accessible outside the class or by the object of the class.Whereas public members such as price and get() and put() functions are accessible by both.

Example2:-

```
#include<iostream.h>
#include<conio.h>
class student
{
protected:
 int roll_no;
char name[20];
}
};
```

```
class result: public student
{
 int marks;
public:
void accept()
{
  cout<<"Enter the roll no of student:";
  cin>>roll_no;
  cout<<"\nEnter the name of student:";
  cin>>name;
  cout<<"\nEnter the marks of a student:";
  cin>>marks;
}
void display()
{
 cout<<"\n\nStudent data is:";
cout<<"\nRoll number="<<roll_no;
cout<<"\nName="<<name;
cout<<"\nMarks="<<marks;
}
};
```

In above example protected members of student base class are inherited in derived class result.

Q.6 (a)(Any two errors-2 marks, Correct Program-2 marks)

```
#include<iostream.h>
    class temp
    {
      private:
       int i;
       float f;
      public:
       tp()
      {
       i=0;
       f=0.0;
      }
   }
   void main()
   {
     tp t1;
   }
```

Following are the errors in above code:

- class should be terminated by a semicolon.
- Undefined symbol tp.
- Statement missing semicolon(;) line no 16.

The correct program is:

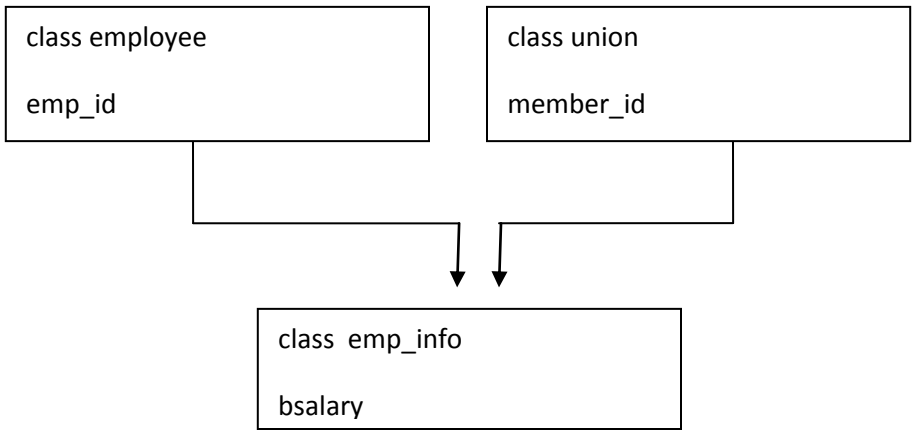
```
#include<iostream.h>
class temp
{
```

```
private:
    int i;
    float f;
  public:
    temp()
  {
    i=0;
   f=0.0;
  }
};
void main()
{
  temp t1;
}
```

Q.6 (b) (Definition-2marks, Diagram-2 marks)

1. A class can inherit the attributes of two or more classes. This is known as multiple inheritance.
2. Multiple inheritance allows us to combine the features of several existing classes as a starting point for defining new classes.

Multiple Inheritance



Q.6 (c)(Any four rules, 1 mark for each rule)

1. Only existing operators can be overloaded.New operators can not be created.
2.The overloaded operator must have at least one operand that is of user defined type.
3. We cannot change the basic meaning of an operator i.e. we cannot redefine the plus(+) operator to subtract one value from the other.
4. Overloaded operators follow the syntax rules of the original operators.They cannot be overridden.
5. There are some operators that cannot be overloaded. for e.g.sizeof, . , .* , : : , ?: .
6.We cannot use friend function to overload certain operators (=,( ),[ ],->).However member functions can be used to overload them.
7.Unary operators overloaded by means of a member function, take no explicit arguments and return no explicit values,but those overloaded by means of a friend function, take one reference argument.
8.Binary operators overloaded through a member function take one explicit argument and those which are overloaded through a friend function take two explicit arguments.

Q.6 (d) (Definition-2 marks, Program-2 marks)

1. A pointer for an array points to the base address of an array.

2. A pointer is initialized to the memory address of first element placed inside an array.

3. We can use this base address for performing arithmetic operations on an array.

```
#include<iostream.h>
#include<conio.h>
void add(int *ptr);
void main()
{
 int a[5]={2,1,3,4,5};
 int *ptr;
 clrscr();
 ptr=a;
 add(ptr);
 getch();
}
```

```
void add(int *ptr)
{
cout<<"Array elements are:";
for(int i=0;i<5;i++)
{
cout<<*ptr<<" ";
ptr++;
}
}
```

Q.6 (e)(Logic-2 marks, Syntax-2 marks)

```
#include<iostream.h>
#include<conio.h>
float volume(float l)
{
return(l*l*l);
}
float volume(float r,float h)
{
return(3.14*r*r*h);
}
void main()
{
float length,radius,height,vol;
clrscr();
cout<<"\n Enter length of cube : ";
cin>>length;
vol=volume(length);
cout<<"\n Volume of Cube : "<<vol;
```

```
 cout<<"\n\n Enter Radius of cylinder : ";
 cin>>radius;
 cout<<"\n Enter height of cylinder : ";
 cin>>height;
 vol=volume(radius,height);
 cout<<"\n Volume of Cylinder : "<<vol;
 getch();
 }
```

Q.6 (f) (Logic-2 marks, Syntax-2 marks)

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=2;i<=4;i+=2)
{
cout<<setw(1)<<i;
cout<<setw(5)<<setfill('-')<<i*2;
cout<<"\n";
}
for(i=6;i<=10;i+=2)
{
```

```
cout<<setw(1)<<i;

cout<<setw(6)<<setfill('-')<<i*2;

cout<<"\n";

}

getch();

}
```

**********END**********