Name - Aman Yadav
Class - B.Sc.Cs I^st. sem
Roll no - 20207005
Subject - Programming fundamental using C++
no. no - 7000620333
email ID - amanyadav592002@gmail.com

Date - 15/04/2021

Answer No - 1.

Class and object - (1) class is a blueprint or template from which objects are created. Object is an instance of class

(2) Class is a logical entity. Object is a physical entity.

(3) object allocates memory. Class doesn't allocate memory when it is created.

inheritance and polymorphism

Inheritance is that in which new class is created that inherits the property of already exists class. It support the concept of reusability and reduce the length of code.

Polymorphism is that in which we can perform a task in multiple way, word polymorphism means many forms.

Abstraction and Encapsulation

1) Abstraction is the process or method of gaining information, while encapsulation is the process of storing or containg information.

2) Abstraction is the method of hiding the unwanted information while in encapsulation it is method to hide the data in a single entity or unit along the method to protect information from outside.

Dynamic binding and Message passing

1) dynamic binding is a method of linking a procedure call to the relavant code that will be executed only on runtime, while message passing is method of exchanging message between object and object and object oriented progsamming.

Answer NO - 2

In C# there are 3 types of loops - (1) while (2) do-while (3) for.

(1) while - while is the loop where we can give a condition if condition is satisfied then it will be executed else it will not.

```
int main()
{
  int a=0;
  while(a<10){
   cout<<"hi"<<endl;
    a++;
  }
  return 0;
}.
```

example of while loop.

(2). do-while loop.

in do-while loop we have one condition to stop the iteration which should be in while the do will start the loop from the keyword do. and stops at infinitely sum from while.

```
int main(){
    int a=4;
    do
     {
       cout<<"hi"<<endl;
       a++;
     }
     while(a<6);
     return 0;
   }
```

(3). for loop - for loop is used when we want to iterate a condition for soo many times in for loop we have to give value from where to start and where to end and also we have to describe the incrementation or decrementation.

```
for(i=0; i<4; i++){
        cout<<"i";
    }
```

output: 0123

Answer No-3

friend Class: a friend class in c++ can access the protected and public member of ~~main~~ base class. we have to declare the other class as friend to give access to the protected and private members.

```
class box{
        private:
        int length;
        public:
        box():length(){}
        friend int printlength(box);    //friend function
};
int printlength(box b)
{
  b.length+=10
   return b.length;
}
```

```
int main(){
      box b;
    cout<<"length of box:"<<printlength(b)<<en
      return 0;
   }
```

output: length of box = 10

Answer No-05

constructor : constructor is a member function of a class that has the same name as the class. It help to initialize object along with class. can accept arguments.

destructor : destructor is also a member function of a class that has the same name as the class preceded by ~(tilde) operator. It is called while object is freed.

```
class z {
    public:
    z() // constructor
     {
      cout<<"constructor called";
     }
    ~z()  // destructor
     {
      cout<<"destructor called";
     }
};
int main(){
     z z1;
    int a=1;
    if(a==1)
     {
      z z2;
     }
}
```

Answer No-6

inline function is a powerful concept that is commonly used with class. the compiler places copy of the code of that function at each point where the function is called at compile time. we have to use key word inline to use this function.

advantage: it doesnot require function overcalling head.
- it also save overhead of the return call from function

disadvantage: may increase function size so that it can't store in cache
- if used in header file it will make your header file may large and unreadable

```
inline int max(int x, int y){
    return (x>y)?x:y;
}
int main()
{
  cout<<"max(20,10):"<<max(20,10);
  return 0;
}
```

output: max(20,10): 20

Answer No-7

concept of oops — it is a programming concept that work on principle of abstraction, encapsulation, inheritance and polymorphism. It allow user to create object and create method to handle those object. The basic concept is to use them and get result

benifits - (1) reusability, data redundancy
(2) code maintance
(3) security
(4) polymorphism flexibility
(5) Problem solving.

Answer No · 8

there is no such thing as static class. but a class can store static members. and methods. static data member in class is shared by all the class objects. as there is only one copy of them in the memory, regardless of number of object of class.

example of static class ⟶

```
class example {
    public:
    static int a;
    static int func(int b){
        cout<<"static member function called\n";
        cout<<"The value of b is : "<<b<<endl;
    }
};
int example::a = 28
int main(){
    example obj;
    example::func(8);
    cout<<"In the value of static data member is : "<<obj.a;
    return 0;
}
```

output: static member function called
the value of b is : 8
the value of static data member is : 28

Answer No - 9

factorial of a number using recursion

```
#include <iostream>
using namespace std;
int factorial(int n){
        int ans = 1;
        for (int i=n; i>0 ; i--)
          ans = ans * i;
          return ans;
}
int main(){
    int no;
    cout<<"enter no. to get factorial:";
    cin>>no;
    cout<< no <<"! = " <<factorial(no)<<endl;
    return 0;
}
```

output: enter no. to get factorial : 5
5! = 120

Answer No- 10

variable: a variable is the name given to a memory location. It is the basic unit of storage in a program. The value stored in a variable can be changed during execution of program.

type conversion: implicit type conversion also known as automatic type conversion. done by the compiler on its own without any external trigger from the user. generally take place when in an expression more than one data type is present.

datatypes: data types are declaration for variables. this determine the type and size of data associated with variable. example — int age = 13;

Operators: operator is the symbom that perform operation on variable and value. for example + is an operator used for addition. while - is an operator used for substraction. there are 8 types of operator.

Answer No - 12

```
#include <iostream>
using namespace std;
int main() {
 int n;
 cout<<"Enter the number of rows :";
 cin>> n;
  for(int i=1,k=0; i<=n ; i++, k=0){
      for(int s=1; s<=(n-i); s++)
          {
              cout<<" s" cout<<" ";
          }
      while(k != (2*i)-1)
       {
        cout<<"*";
        k++;
       }
       cout<<endl; }
      return 0;
}
```

output:

```
Enter number of rows : 4
   *
  ***
 *****
*******
```

Answer No-13

- like array of other user defined data types an array of type class can be created. this array class contains the object of class as its individual elements.
- thus and array of a class type is also known as array of objects

syntax : class-name array-name[size];

```
class my_class {
    int x;
    public:
      void set(int i) {x=i;}
      int getx() {return x;}
};
void main() {
      my class obj[4];
    int i;
    for(i=0; i<4; i++){
      obj[i].setx(i); }
    for(i=0; i<4; i++){
      cout << " obs[" << i << "].getx() : " << obj[i].getx() << endl;
    return 0;
}
```

output:

```
obs[0].getx() : 0
obs[1].getx() : 1
obs[2].getx() : 2
obs[3].getx() : 3
```

Answer No-14

in C++, operator overloading, is a case of polymorphism, where different operators have different implementation depending on their arguments it overload the same operator but with different argument.

example : + operator

7+8 = 15 // here + operator will give addition of two digit

"str" + "ing" = string // here + operator used to concatenate strings

0.15 + 9.25 = 9.40 // here + operator used for floating point addition.

Basic rules ① only existing operators can overload.

② overload operator must have atleast one operand of user defined type.

③ we can't change basic meaning of operator.

④ overloading operators follow the syntax rules of original operators.

Practical No - 16

Virtual function : A virtual function is a member function which is declared with in a base class and is overridden by derived class. when derived class object using a pointer or refrence to base class you can call virtual function of that object and execute its derived class's version.

early binding :- also known as static binding, an object is performed to a variable declared to be a specific object type. early binding are basically strong type of object. this are checked during compile time.

late binding :- here methods and process are checked only at the runtime. It implies that compiler doesnot know what kind of object or which method is contains until runtime advantage is that it can hold refrence to any object.

```
class base {
        public:
        void show() { cout<<"in base ";}
};
int main
class derived : public base{
        public:
            void show() { cout<<" in derived ";}
};
int main() {
            base *bp = new derived;
            bp->show();
            return 0;
}
```

output: in base

Answer No -18

```
#include <iostream>
using namespace std;
int search (int arr [], int n, int x)
{ int i ;
  for (i=0 ; i<n ; i++ )
      if (arr [i] ==x)
        return i ;
    return -1 ;
}
int main (){
    int array [] = {2,3, 4,10, 40};
    int x = 10 ;
    int n = sizeof(array)/sizeof(array [0]) ;
    int result = search (arr, n, x) ;
      (result==-1)
        ? cout<<"element is not present in array " : cout<<"element is present
    at index "<<resullt ;
        return 0 ;
}
```

output :

element is present at index 3

Answer No - 19

```
#include <iostream.h>
#include <conio.h>
#include <dos.h>
void main () {
      struct student
      {
        int m ;
        int sub [10] ;
      } ;
      struct student st [50] ;
      int i,j,n, m, total ;
      float av ;
      clrscr() ;
      cout<< "enter the number of student : " ;
      cin >> n ;
      cout << "enter the number of subject taken : " ;
      cin >> m ;
      for(i=0 ; i<n ; i++)
       { total = 0
       cout<< "enter roll of student "<< i+1 << "student" ;
       cin>> st [i].rn ;
continuing from bottom of left
       cout<<"\n enter the marks : " ;
       for (j=0 ; j<m ; j++ ){
         cout<< "enter the marks of "<<j+1 <<"
       subject : " ;
          cin>>st [i].sub [j] ;
          total = total + st[i].sub[j] ;
        }
          av =(float) total / m ;
        cout << "average mark of " << i+1 <<
        "student = " << av ;
          }
           getch() ;
      }
```