Assignment - 2

Q.1 Local variables are stored in an area called ____

Ans stack

Q.2
```
#include <iostream>
using namespace std;
class Base();
class Derived : public Base();

int main()
{
    Base *bp = new Derived;
    Derived *dp = new Base;
}
```

Ans Compiles error in line "Derived *dp = new Base;"

Q.3 When the inheritance is private; the private methods in base class are _____ in the derived class (in C++).

Ans Inaccessible.

Q.4 Which of the following is true?

Ans a) The number of times destructor is called depends upon number of objects created.

Q.i State true or false.
Type Conversion is automatic where as type casting is explicit.

Ans True.

Short answer type question?

Que 1. Explain about new and delete Keywords with code.

Ans Dynamic memory allocation means. creating memory at run time.

→ If we need to allocate memory at runtime, we must use new operator followed by data type.
If we need to allocate memory for more than one element, we must provide total numbers of elements required in square bracket []. It will return address of first byte of memory.

Syntax 2

```
ptr = new data-type ;
// allocate memory for one element //

ptr = new data-type [size];
// allocate memory for fixed no. of element //
```

→ Delete operator is used to deallocate the memory created by new operator new- operator at run-time. Once the memory is no longer needed it should by freed so that the memory becomes available again for other request of dynamic memory.

Syntax: delete ptr;
// deallocate memory for one element //

delete [ ] ptr;
// deallocate memory for array //

Qu 2. What are constructors? Why are they required? Explain different types of constructor with suitable example.

Ans: A Constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object create. It is special member function of the class.

→ Constructor initialize the new object. that is, they setup the startup property values for the object. They might also do other things necessary to make the object usuable.

Syntax: ClassName ( ) {
}

* Types of Constructor:

① Default Constructor:
It is the constructor which doesn't take any argument. It has no parameters.

Eg: public class Adminclass
{
String userId = String.Empty;
String password = String.Empty;

```
public Adminclass()
  {
    userId = "abi";
    password. "xyz";
  }
}
```

(2) Parameterized Constructor :

This constructor is created by the developer. This constructor takes ~~place~~ atleast one parameter.

For eg:

```
public class Adminclass
  {
    String userId = String.Empty;
    String password = String.Empty;

Public Adminclass (String username, String userpassword)
  {
    userId = username;
    password = userpassword;
  }
}
```

(3) Copy Constructor :

A copy constructor is a member function which initializes an object using another object of the same class.

For eg:

```
class point
{
  private:
```

```
    int x, y;
public:
    point ( int x1, int y2)
{
    x = x1;
    y = y2;
}
    point ( const point & obj )
{
    x = obj.x;
    y = obj.y;
}
}
```

(9) Static Constructor :-

It is used to initialize static fields. It can also be used to perform any action that is to be performed only once. It is invoked automatically before first instance is created.

for eg:

```
Public static class Calculator
{
    private static int resultStorage = 0;
    public static string Type = "Arithematic";
    public static int Sum (int num1, int num2)
    {
        return num1 + num2;
    }
    public static void store (int results)
    {
        resultStorage = results;
    }
}
```

(1) Private Constructor:

Special Instance constructor is a private constructor. It is generally used in classes that contain static members only.

Ex:

```
public class sample
{
  private sample()
  {
    Console.WriteLine("Private Constructor");
  }
}
  class sample1
{
  static void Main()
{
  sample obj = new sample1();
}
}
```

Ques 3: Explain the difference between object oriented and procedural programming language in detail?

Ans:

Both Procedural programing and object oriented programming are the high level languages in programming and are widely used in development of applications. On the basis of nature of developing the code both languages have different approaches on basis

of which both are differentiate from eachother.

Key	(OOP)	(POP)
Definition	→ OOP is a programming language that uses classes and objects to create models based on the real world environment.	→ POP is a programming language that follows a step by step approach to break down a task into a collection of variables and routines through a sequence of instructions.
Approach	→ In OOPs, concept of objects and classes is introduced and hence the program is divided into small chunks called object	→ In POPs, the main program is divided into small parts based on the functions and is treated as separate program for individual smaller program.
Access modifiers	→ In OOPs, access modifiers are introduced namely as a "Private", "public".	→ No such modifiers are introduced in POP.
Security	→ Due to abstraction in OOPs data hiding is possible and hence it is more secure than POP.	→ POP is less secure as compare to OOPs.

Long Answer Type Question:

Q.) Explain the type of polymorphism with code.

Ans Types of polymorphism:

(1) Compile Time
* Function overloading
* operator overloading

(2) Runtime
* Function overriding
* Virtual function

(1) Compile time polymorphism:

In this, a function is called at the time of program compilation. We call this type of polymorphism as early binding or static binding.

* Finding Overloading:

Function overloading means one function can perform only one task. In C++, a single function is used to perform many tasks with the same name and different types of arguments. In function overloading function will call at the time of program compilation. It is an example of compile-time polymorphism.

For eg:

```
class Addition {
public:
    int ADD (int x, int y)
    {
        return x+y;
    }
```

```
int ADD()
{
  string a = "Hello";
  string b = "SAM";
  string c = a+b;
  cout << c << endl;
}
};
  int main(void)
{
  Addition obj;
  cout << obj.ADD(120, 3) << endl;
  obj.ADD();
  return 0;
}
```

* Operator Overloading :-

Operator overloading means defining additional task to operators without changing its actual meaning. The purpose of Operator overloading is to provide a special meaning to the user-defined data types. Advantage of operators overloading is to perform different operations on the same operand.

For eg:-

```
class A
{
  string x;
  public:
```

```
A () { }
A ( string i)
{
  x = i;
}
void operator +(A);
  void display ();
};
  void A :: operator +(A, a)
{
  string m = x+a.x;
  cout<<" The result of the addition of two objects is
  : "<<m;
}
  int main ()
{
  A a1 ("welcome");
  A a2 ("back");
   a1+a2;
  return 0;
}
```

* 2. Runtime Polymorphism :-

In this, functions are called at the time the program execution. Hence, it is known as late binding binding.

Function overriding is a part of runtime polymorphism. In function overriding, more than one method has the same name with different types of the parameter list.

It is achieved by using virtual functions and

pointers. It provide slow execution as it is known at the sun time. Thus, it is more flexible all the things executed at the sun time.

* Function overriding:

In this, we give the new definition to base function in the derived class. At that time, we can say the base function has been overridden. It can be only possible in 'derived class'. In function overriding, we have two defitions of the same function. one in the super class and one in derived class. The decisions about which function definition requires calling happens at suntime. That is the reason we called it suntime polymorphism.

for eg

```
Class Animal
{
public:
void function()
{
Cout << " Eating ... " << endl;
}
};
class Man: public Animal
{
public:
void function ()
{
```

```
    cout << "walking " << endl;
  }
};
  int main (void)
{
  Animal A = Animal ();
   A. function ();
   Man m = Man();
  m. function ();
  return 0;
}
```

* Virtual ~~Prog~~ Functions:

A virtual function is declared by keyboard virtual. The return type of virtual function may be Int, float, void.
A virtual function is a member function in the base class. We can redefine it in derived class. It is part of runting polymorphism. The declaration of the virtual function must be In the base class by using the keyboard virtual. A virtual function is not static The virtual function helps to tell the compiler to perform dynamic binding or late binding on the function.

For eg:-

```
class Add
{
  Int x=5, y=20;
  public:
  void display ()
{
```

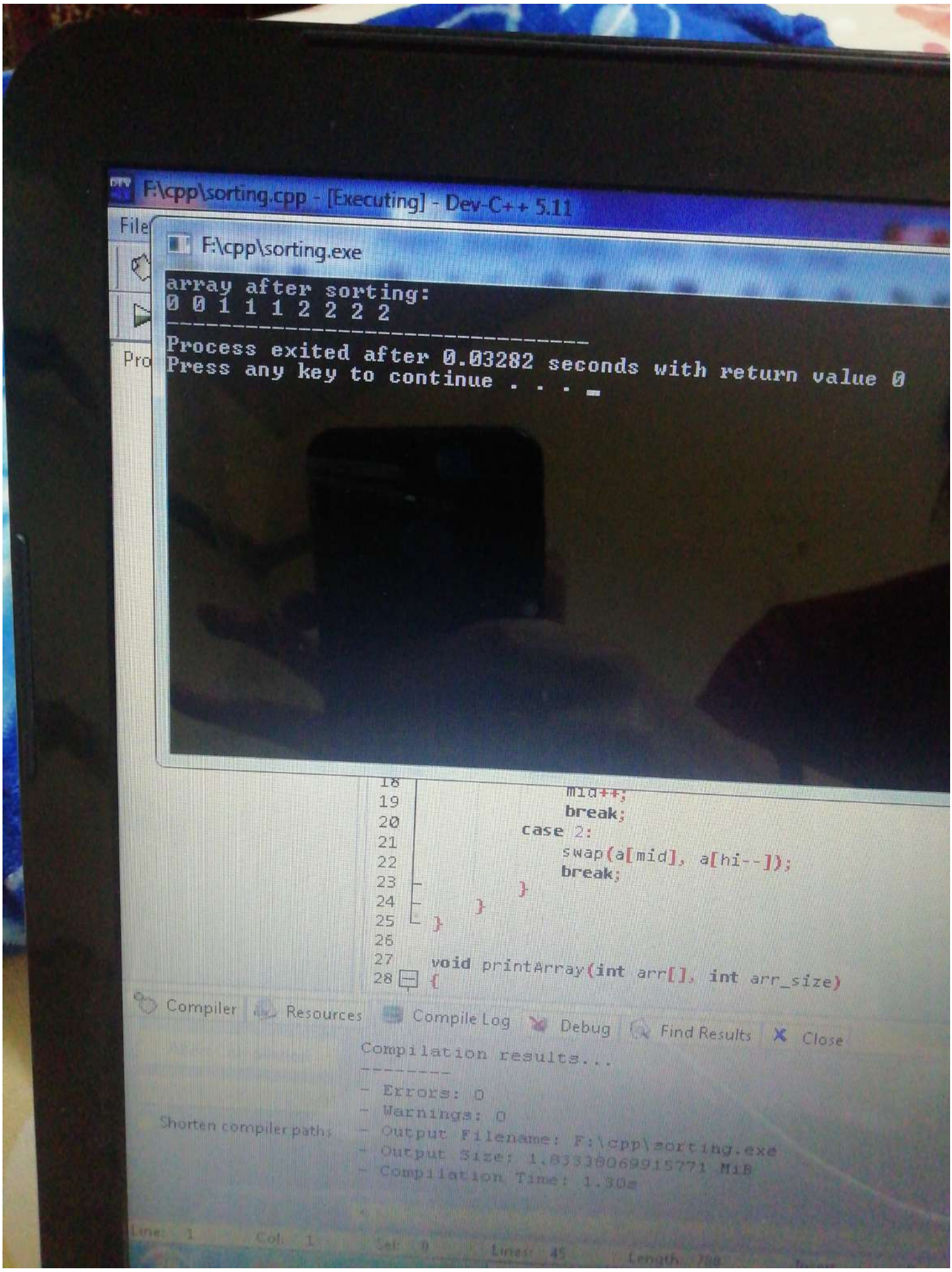
```
    cout << " value of x is : " << x+y << endl;
  }
};
  class subtract :
      public Add.
    int y =10 , z=30;
  class Add
  {
    int x=5 , y=20;
    public :
     void display ()
    {
     cout << " value of x is : " << x+y << endl;
   }
};
  Class subtract :
      public Add.
  {
     int y=10; z=30;
     public:
   void display ()
{
   cout << " value of y is :" << y-z << endl;
 }
};
    int main()
   {
     Add *m;
     subtract s;
     m = &s;
     m -> display ();
   } return 0;
```

Q8

```
#include <bits/stdc++.h>
using namespace std;
void sort(int a[], int arr_size)
{
    int lo = 0;
    int hi = arr_size - 1;
    int mid = 0;

while (mid <= hi){
    switch (a[mid]){
    Cas 0;
        swap(a[lo++], a[mid++]);
        break;
    Case 1;
        mid++;
        break;
    Case 2:
        swap(a[mid], a[hi--]);
        break;
    }}}
Void print Assay (int arr[], int arr_size)
{
    for (int i = 0; i < arr_size; i++)
        cout << arr[i] << " ";
}
int main()
{
    int arr[] = {1, 1, 2, 2, 0, 0, 2, 1, 2};
int n = size of (arr) / size of (arr[0]);
sort (arr, n);
    Cout << "Assay after sorting : " << endl;
```

```
printArray(arr, n);
    return 0;
}
```


F:\cpp\sorting.cpp - [Executing] - Dev-C++ 5.11
File
F:\cpp\sorting.exe
array after sorting:
0 0 1 1 1 2 2 2 2

Process exited after 0.03282 seconds with return value 0
Press any key to continue . . .
mid++;
break;
case 2:
swap(a[mid], a[hi--]);
break;
}
}
}
void printArray(int arr[], int arr_size)
{
Compiler
Resources
Compile Log
Debug
Find Results
Close
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: F:\cpp\sorting.exe
- Output Size: 1.83380699157715 MiB
- Compilation Time: 1.30s
Shorten compiler paths

using name space std;

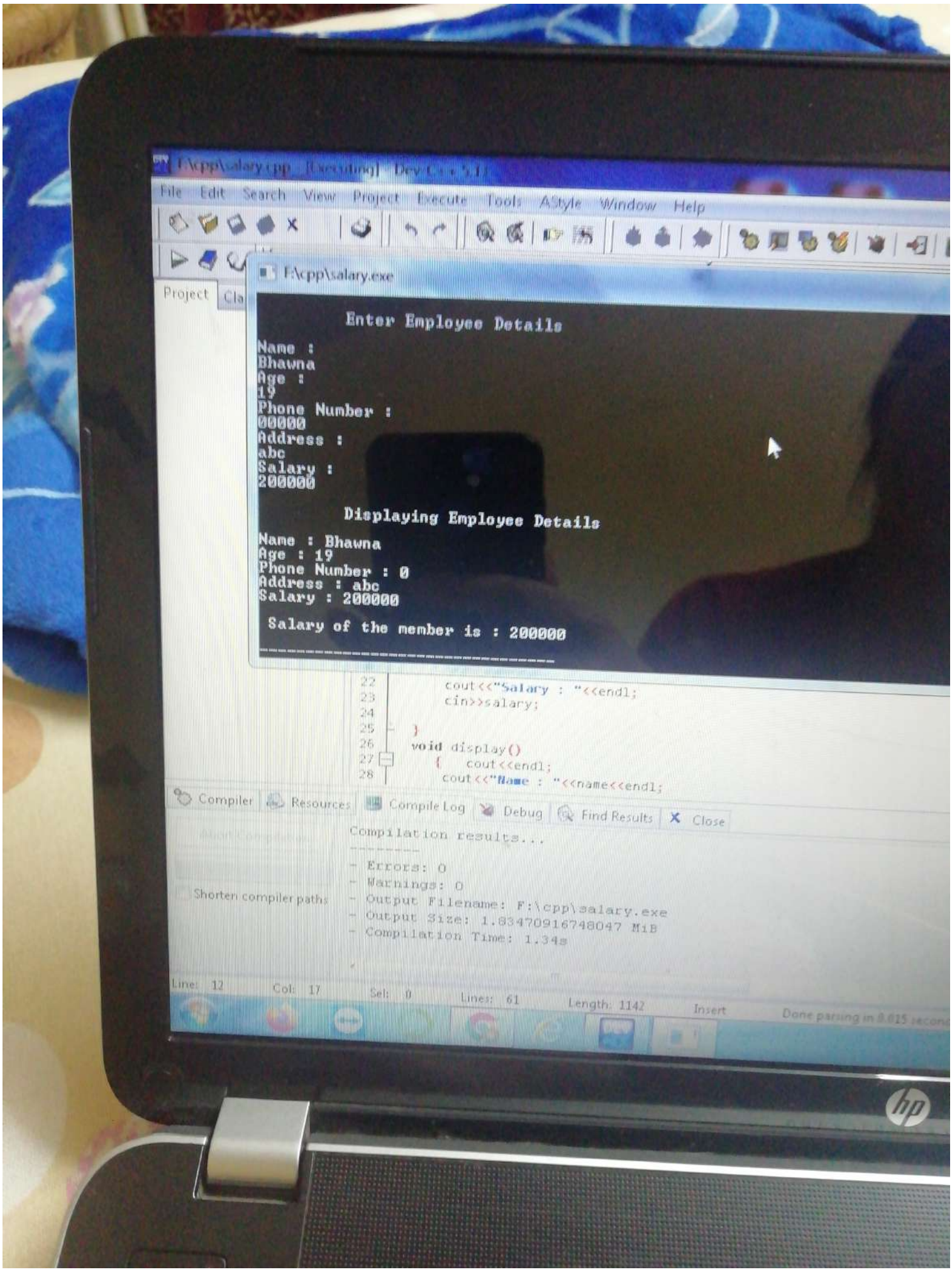
Q. Ans

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

class member {
char name[20], address[40];
  double number;
  int age;

public:
  int salary;
  void input()
{
  cout << endl;
  cout << "Name :" <<endl;
  cin >> name;
  cout << "Age :" <<endl;
  cin >> age;
  cout << "Phone number :" <<endl;
  cin >> number;
cout << " Address:" <<endl;
  cin >> address;
cout << " Salary :" endl;
  cin >> Salary;
}
```

```
Void ~~salary~~ display()
 {
   cout<<endl;
   Cout<<"Name.:"<<name<<endl;
   Cout<<"Age:"<<age<<endl;
   Cout<<"Phone number:"<<number<<endl;
   Cout<<"Address:"<<address<<endl;
   Cout<<"Salary:"<<salary<<endl;
 }
};
  class employee : public member
{
  public:
    void input()
{
  ~~input~~ cout<<"\n\t Enter Employee Details \t\n";
  member::input();
}
  void display()
  {
    Cout<<"\n\t Displaying Employee Details \t\n";
  member::display();
}
  void getsalary()
  {
    Cout<<"\n salary of the member is:"<<salary<<
  }                                          endl;
};
    int main(){
    employee e;
    e.input();
    e.display(); e.printsalary(); }
```


F:\cpp\salary.exe
Enter Employee Details
Name :
Bhawna
Age :
19
Phone Number :
00000
Address :
abc
Salary :
200000
Displaying Employee Details
Name : Bhawna
Age : 19
Phone Number : 0
Address : abc
Salary : 200000
Salary of the member is : 200000
File Edit Search View Project Execute Tools AStyle Window Help
Compiler Resources Compile Log Debug Find Results Close
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: F:\cpp\salary.exe
- Output Size: 1.83470916748047 MiB
- Compilation Time: 1.34s