

I. choose the correct option:-

Q1 Local variables are started in an area called Stack

- (a) Heap (c) Free memory
(b) permanent (d) stack

Q2 Choose the correct option?

```
#include using
namespace std;
class Base {
class Derived : public
Base {
int main()
{
```

```
Base *bp = new
```

```
Derived,
```

```
Derived *dp = new
```

```
Base; }
```

- (a) NO Compiler Error
(b) Compiler Error in line "Base

(c) Compiler Error in line Derived
*dp = new Base

(d) Runtime Error

Q3 When the inheritance is private, the private methods in base class are Inaccessible in the derived class C++

- (a) In accessible (c) Protected
(b) Accessible (d) public

Q4 which of the following is true ?

- (a) The number of times destructor is called depends on number of objects created.
- (b) Destructor is called only once
- (c) there can be more than one destructors at the class.
- (d) programmer have always call destructor at the end of the program.

Q5 state true or False

- (a) Type Conversion is automatic whereas type casting is explicit.
- (b) True
- (c) False

II. Short Answer type question:-

Q1 Explain about new and delete keywords of code

Ans: Syntax to use new operator: To allocate memory of any data type, the syntax is:-

`pointer:variable = new data-type;`

delete operator is use to dellocate the memory user privilege to deallocate the created pointer variable by this delete operator

Syntax:- `// Release memory allocation pointed by pointer variable`
`delete pointer-variable`

lets topic a example:-

Allocation and deallocation of memory using new and delete

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
int *p = NULL;
```

```
p = new (nothrow) int;
```

```
if (!p)
```

```
cout << "allocation of the memory failed\n";
```

```
else
```

```
*p = 80;
```

```
cout << "value of p: " << *p << endl;
```

```
}
```

```
float *x = new float (12.86);
```

```
cout << "value of x: " << *x << endl;
```

```
int n = 8;
```

```
int *q = new (nothrow) int (n);
```

```
if (!q)
```

```
cout << "allocation on the memory failed\n";
```

```
else {
```

```
for (int i = 0; i < n; i++)
```

```
q[i] = i+1;
```

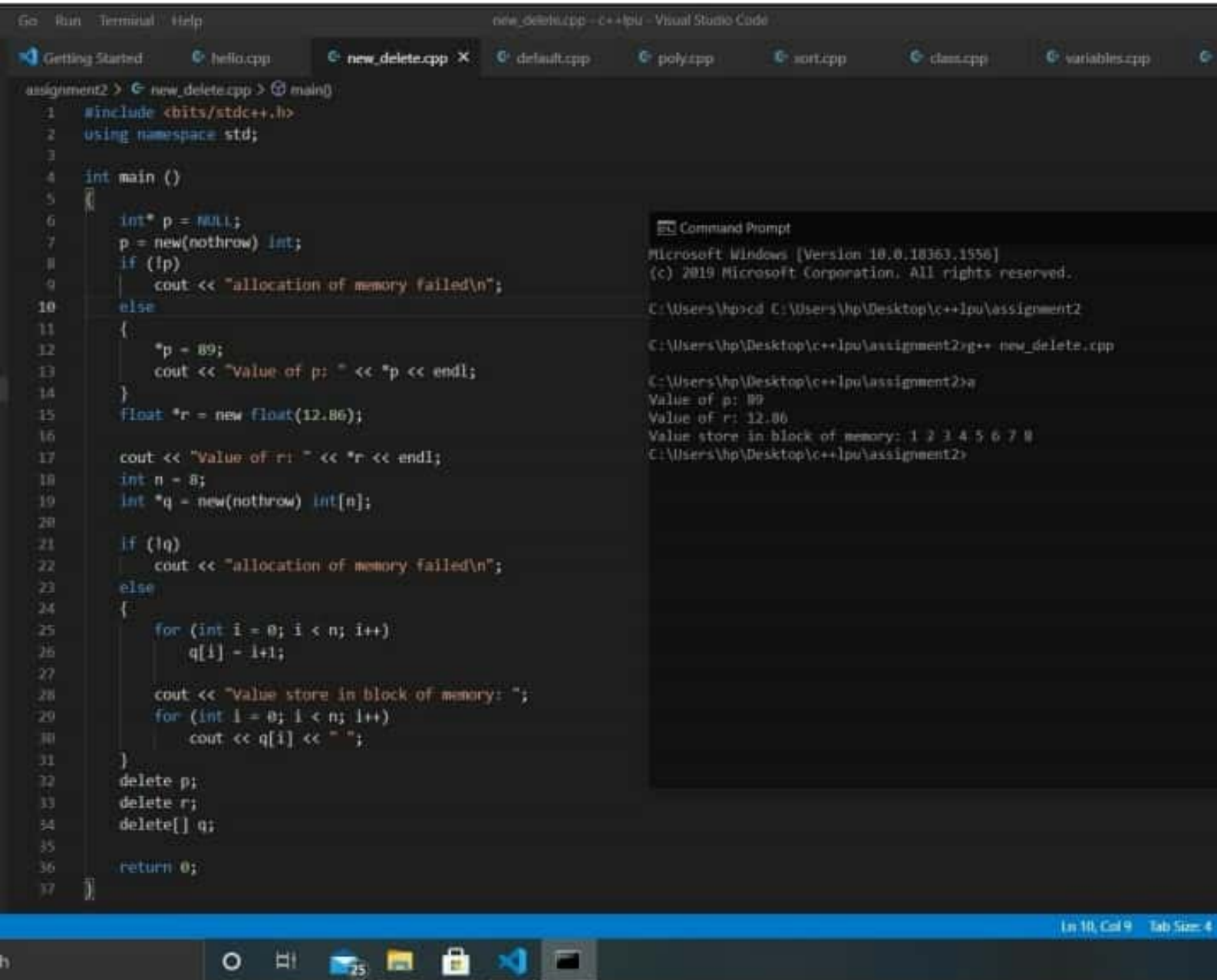
```
cout << "value store in block of memory: ";
```

```
for (int i = 0; i < n; i++)
```

```
cout << q[i] << " ";
```

```
}
```

/ & delete operator:-



The image shows a Visual Studio Code editor with a C++ file named `new_delete.cpp`. The code demonstrates the use of `new` and `delete` operators for memory management. It includes a `main` function that declares and allocates memory for an integer pointer `p`, a float pointer `r`, and an integer array `q`. The program prints the values stored in memory and then uses `delete` to free the allocated memory.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main ()
5 {
6     int* p = NULL;
7     p = new(nothrow) int;
8     if (!p)
9         cout << "allocation of memory failed\n";
10    else
11    {
12        *p = 89;
13        cout << "Value of p: " << *p << endl;
14    }
15    float *r = new float(12.86);
16
17    cout << "Value of r: " << *r << endl;
18    int n = 8;
19    int *q = new(nothrow) int[n];
20
21    if (!q)
22        cout << "allocation of memory failed\n";
23    else
24    {
25        for (int i = 0; i < n; i++)
26            q[i] = i+1;
27
28        cout << "Value store in block of memory: ";
29        for (int i = 0; i < n; i++)
30            cout << q[i] << " ";
31    }
32    delete p;
33    delete r;
34    delete[] q;
35
36    return 0;
37 }
```

The terminal window on the right shows the output of the program:

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++\lpu\assignment2

C:\Users\hp\Desktop\c++\lpu\assignment2>g++ new_delete.cpp

C:\Users\hp\Desktop\c++\lpu\assignment2>a
Value of p: 89
Value of r: 12.86
Value store in block of memory: 1 2 3 4 5 6 7 8
C:\Users\hp\Desktop\c++\lpu\assignment2>
```



```

delete p;
delete x;
delete [ ] q;
return 0;

```

OP :- Value of p = 89
 Value of x = 12.89
 Value store in block of memory:
 12345678

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

Ans. A constructor is a member function of a class with the same name as the class. In C++ constructor is automatically called when object (instance of class) is created. It is a special member function of class. The main purpose of the class constructor in C++ programming is to construct an object of a class. In other words, it is used to initialize all class data members.

Types of Constructors:-

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

Syntax:- class name (parameter1, parameter2, ...)

```

{ // constructor definition
}

```

Example:-

```
#include <iostream>
```

```
using namespace std;
```

```
class Cube
{

```

```
public

```

```
int side;
```

```

cube C;
{ side = 10;
  }
int main()
{
  cube C;
  cout << C.side;
  } return 0; }
O/p → 10

```

2. parameterized constructor:- These are the constructors with parameter. Using this constructor you can provide different values to data members of different objects by passing the appropriate values as argument.

Example:-

```
#include <iostream>
```

```
using namespace std;
```

```
class cube
```

```
{ public
```

```
int side;
```

```
cube (int x)
```

```
{
```

```
side = x;
```

```
} }
```

```
int main ()
```

```
{
```

```
cube C1 (10);
```



```

Cube C2 (20);
Cube C3 (30);
Cube C4 (40);
cout << C1 Side;
cout << C2 Side;
cout << C3 Side;
return 0;

```

Output :-	10
	20
	30

(3) Copy constructor:- It is used in create a copy of an already existing object of class type. It is usually of the form $X(X)$, where X is the class name. The compiler provides a default copy constructor to all the classes.

Syntax:-

```

class name (const class name & object name)
{
    .....
}

```

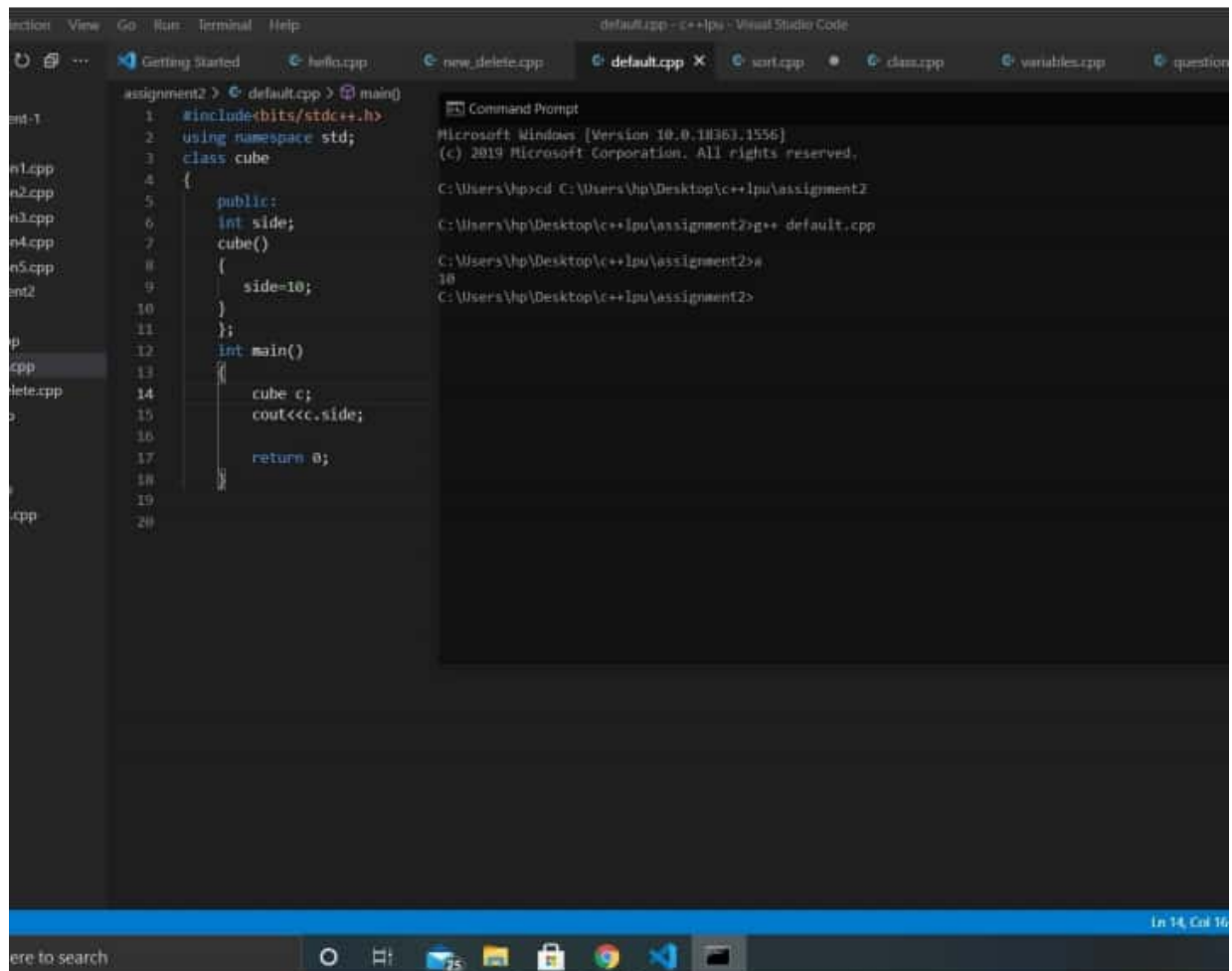
Example:-

```

#include <bits/stdc++.h>
using namespace std;
class Sample Copy Constructor
{
private:
    int x, y;
public:
    Sample Copy constructor(int x1, int y1)

```

//default constructor:-



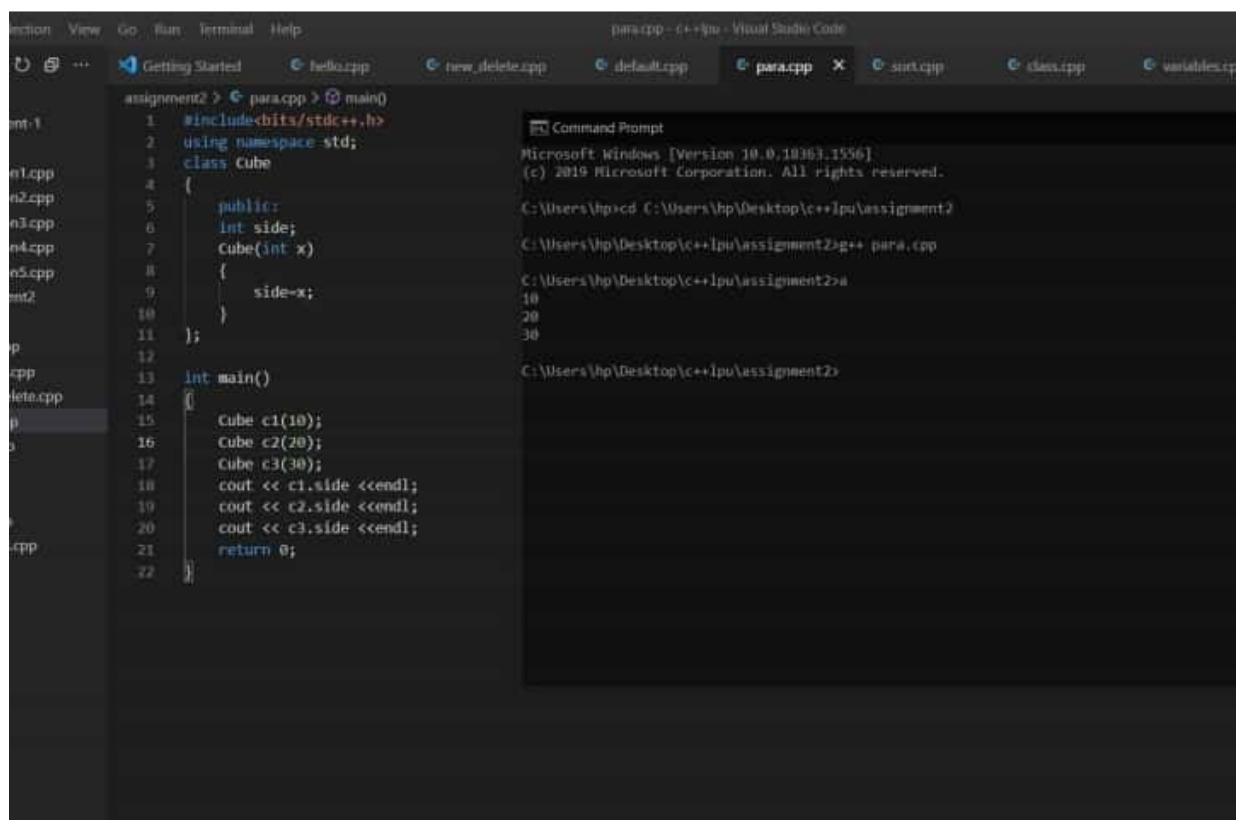
```
assignment2 > default.cpp > main()
1  #include<bits/stdc++.h>
2  using namespace std;
3  class cube
4  {
5      public:
6          int side;
7          cube()
8          {
9              side=10;
10         }
11     };
12     int main()
13     {
14         cube c;
15         cout<<c.side;
16     }
17     return 0;
18 }
19
20
```

Command Prompt

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++lpu\assignment2
C:\Users\hp\Desktop\c++lpu\assignment2>g++ default.cpp
C:\Users\hp\Desktop\c++lpu\assignment2>a
10
C:\Users\hp\Desktop\c++lpu\assignment2>
```

//parameterized constructor:-



```
assignment2 > para.cpp > main()
1  #include<bits/stdc++.h>
2  using namespace std;
3  class Cube
4  {
5      public:
6          int side;
7          Cube(int x)
8          {
9              side=x;
10         }
11     };
12
13     int main()
14     {
15         Cube c1(10);
16         Cube c2(20);
17         Cube c3(30);
18         cout << c1.side <<endl;
19         cout << c2.side <<endl;
20         cout << c3.side <<endl;
21         return 0;
22     }

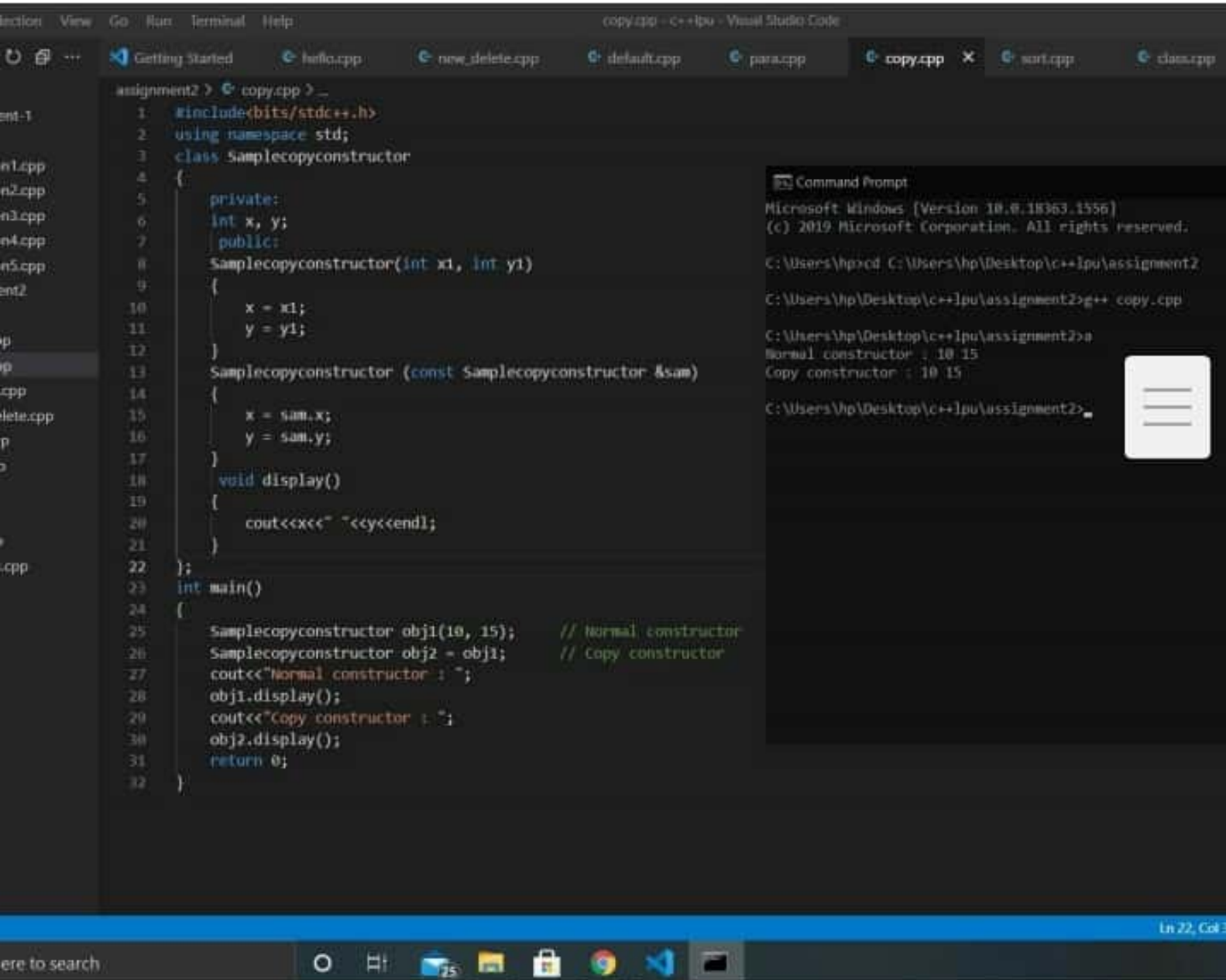
```

Command Prompt

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++lpu\assignment2
C:\Users\hp\Desktop\c++lpu\assignment2>g++ para.cpp
C:\Users\hp\Desktop\c++lpu\assignment2>a
10
20
30
C:\Users\hp\Desktop\c++lpu\assignment2>
```


//copy constructor:-



```
1 #include<bits/stdc++.h>
2 using namespace std;
3 class Samplecopyconstructor
4 {
5     private:
6     int x, y;
7     public:
8     Samplecopyconstructor(int x1, int y1)
9     {
10         x = x1;
11         y = y1;
12     }
13     Samplecopyconstructor (const Samplecopyconstructor &sam)
14     {
15         x = sam.x;
16         y = sam.y;
17     }
18     void display()
19     {
20         cout<<x<<" "<<y<<endl;
21     }
22 };
23 int main()
24 {
25     Samplecopyconstructor obj1(10, 15); // Normal constructor
26     Samplecopyconstructor obj2 = obj1; // Copy constructor
27     cout<<"Normal constructor : ";
28     obj1.display();
29     cout<<"Copy constructor : ";
30     obj2.display();
31     return 0;
32 }
```

Command Prompt

Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++\Ipu\assignment2

C:\Users\hp\Desktop\c++\Ipu\assignment2>g++ copy.cpp

C:\Users\hp\Desktop\c++\Ipu\assignment2>a

Normal constructor : 10 15

Copy constructor : 10 15

C:\Users\hp\Desktop\c++\Ipu\assignment2>

```

{
    X=X1;
    Y=Y1;
}

```

Sample copy constructor (const sample &sample copy constructor)

```

{
    X = Sam.X;
    Y = Sam.Y;
}

```

```

void display ()
{

```

```

    cout << X << " " << Y << endl;
}
}

```

```

int main ()
{

```

```

    Sample copy constructor obj1 (10, 15);

```

```

    Sample copy constructor obj2 (&obj1);

```

```

    cout << "Normal constructor:";

```

```

    obj1.display ();

```

```

    cout << "copy constructor:";

```

```

    obj2.display ();

```

```

    return 0;
}

```

Output:-

copy constructor: 10, 15

Q3 Explain the difference between object oriented and procedural programming language in detail.

Ans → Object-oriented programming

(i) In oop, program is divided into small parts called object

(ii) It follows bottom up approach

(iii) It can access specifier like public, private and protected

Object oriented programming

(iv) Adding new data and function is easy

(v) Object oriented programming provides data hiding & it is more secure.

(vi) Overloading is possible in object oriented programming

(vii) In object oriented programming, data is more important than function.

(viii) It is based on real world

procedural oriented programming

(i) In procedural programming, program is divided into small parts called functions.

(ii) It follows top-down approach

(iii) There is no access specifier in procedural programming

procedural programming

(iv) Adding new data and function is not easy.

(v) Procedural programming does not have any proper way for hiding data & it is less.

(vi) In this programming overloading is not possible.

(vii) In procedural programming, function is more important than data.

(viii) It is based on unreal world

Long Answer type questions:-

Explain the type of polymorphism with code:-

polymorphism :- The word polymorphism means having many forms. polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object instances, the function

Type of polymorphism

Compile time polymorphism

Run time polymorphism

Compile time polymorphism:- This is also known as static overloading. binding function overloading and operators overloading are perfect example of compile time polymorphism.

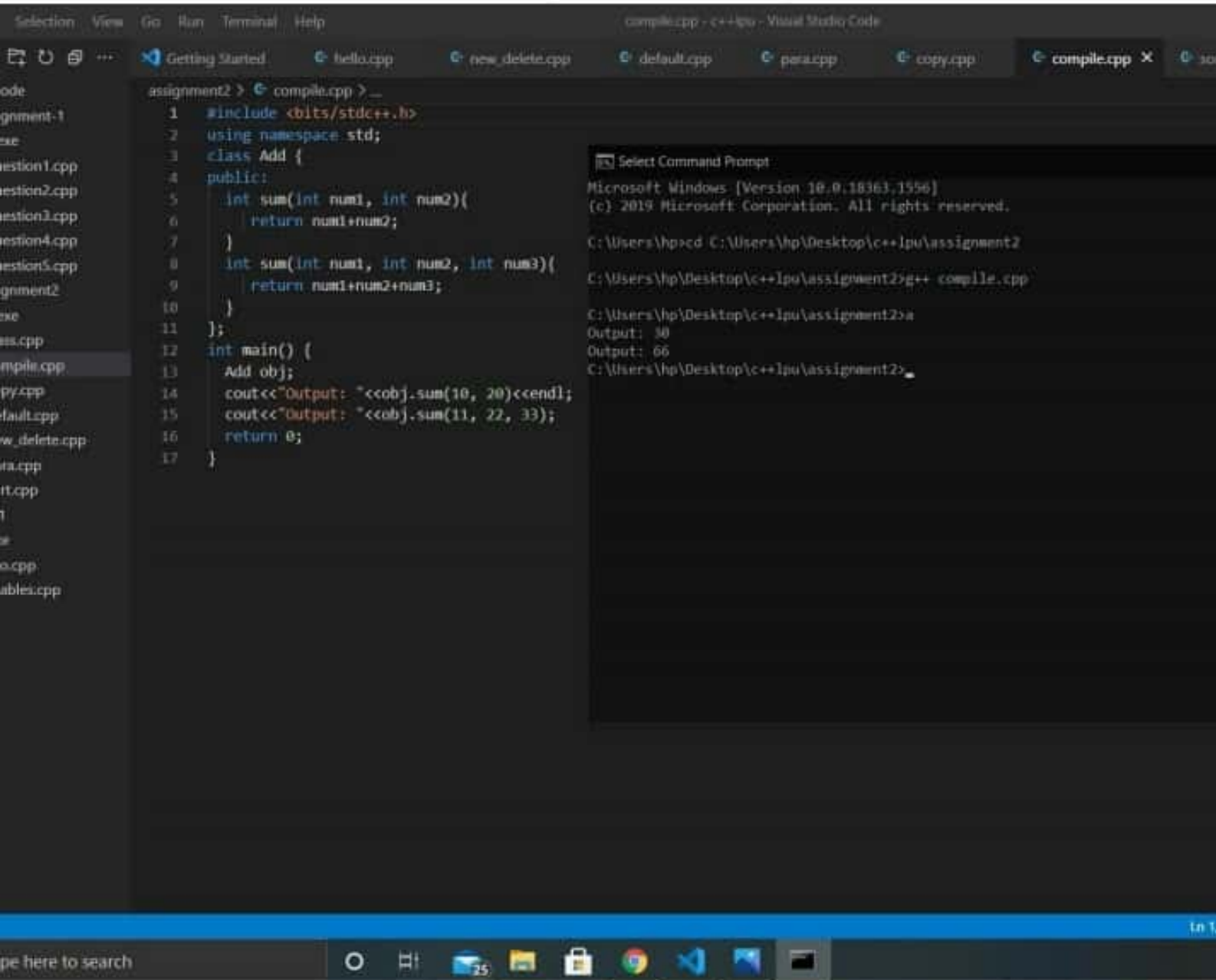
Example:- In this example we have two functions with same name but different number of arguments. Based on how many parameter we pass during functions calls determine which function is to be called. This is why it is considered as an example of polymorphism because in different conditions the output is different since the call is determined during compile time that's why it is called compile time polymorphism.

Code :- #include <iostream>

using namespace std;

class Add {

//compile time polymorphism:-



The image shows a Visual Studio Code editor window with a C++ file named `compile.cpp`. The code defines a class `Add` with two public methods: `sum(int num1, int num2)` and `sum(int num1, int num2, int num3)`. The `main` function creates an object `obj` and calls both methods, printing their results.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 class Add {
4 public:
5     int sum(int num1, int num2){
6         return num1+num2;
7     }
8     int sum(int num1, int num2, int num3){
9         return num1+num2+num3;
10    }
11 };
12 int main() {
13     Add obj;
14     cout<<"Output: "<<obj.sum(10, 20)<<endl;
15     cout<<"Output: "<<obj.sum(11, 22, 33);
16     return 0;
17 }
```

The output of the program is shown in the terminal window:

```
Select Command Prompt
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++lpu\assignment2
C:\Users\hp\Desktop\c++lpu\assignment2>g++ compile.cpp

C:\Users\hp\Desktop\c++lpu\assignment2>a
Output: 30
Output: 66
C:\Users\hp\Desktop\c++lpu\assignment2>
```

Public:

```
int sum (int num1, int num2) {
    return num1 + num2;
}
```

```
int sum (int num1, int num2, int num3) {
    return num1 + num2 + num3;
} ;
```

```
int main() {
```

Add obj;

```
cout << "output: " << obj.sum (10, 20) << endl;
```

```
cout << "output: " << obj.sum (11, 22, 33);
```

```
return 0;
```

```
}
```

O/p:- output : 30

output : 66

(ii) Run time polymorphism :- This is also known as dynamic (or late) binding

Example:- Function overriding is an example of Runtime polymorphism.

Function overriding: when child class declares a method which is already present in the parent class then there is called function overriding here child class overrides the parent class. In case of function overriding we have two definitions of same function one is parent class and one is child class. The call the function is determined at runtime to decide which definition of the function.

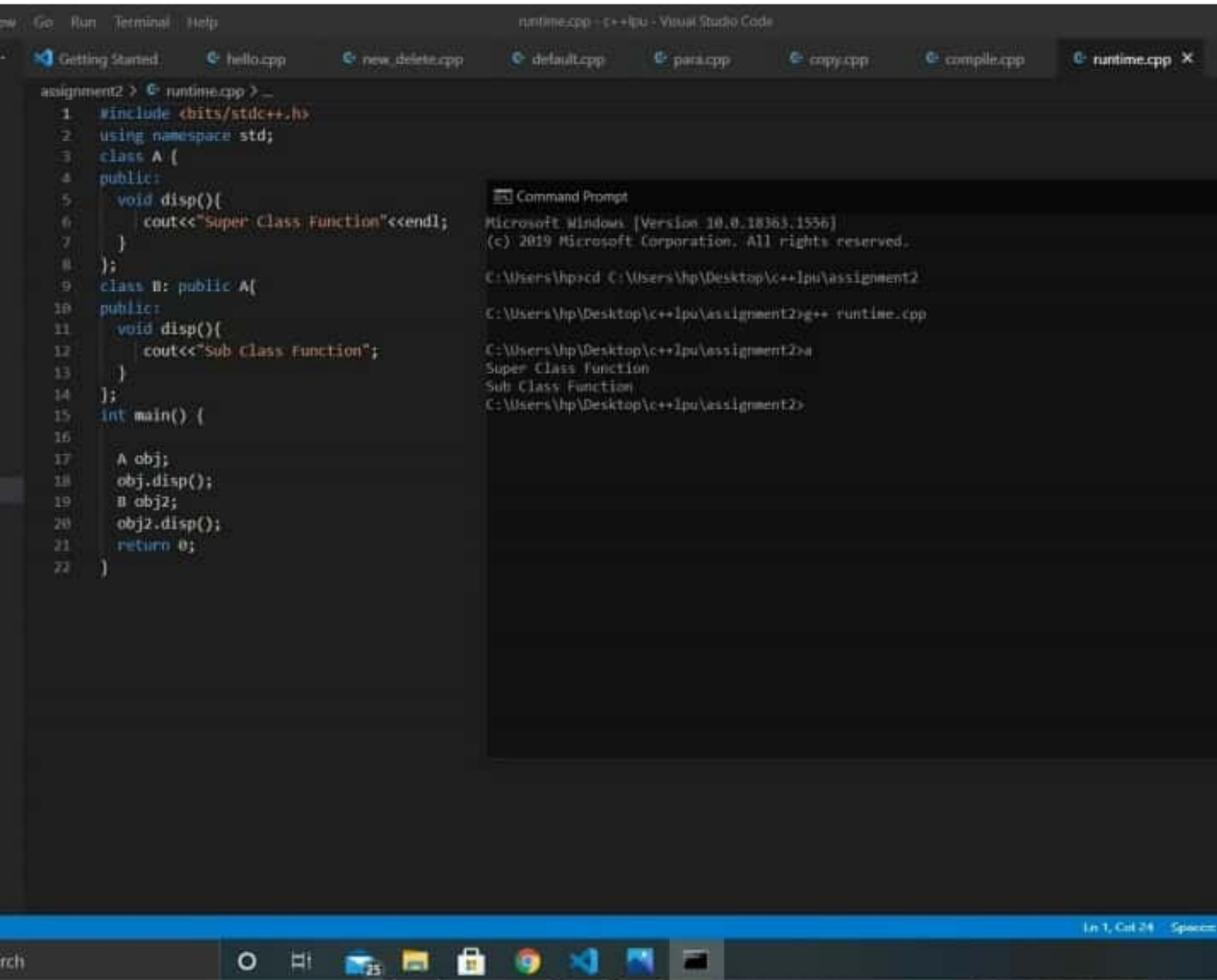
Page: 11

```

Code :- #include <bits/stdc++.h>
        using namespace std;
        class A {
        public:
        void display() {
        cout << "Super class Function" << endl;
        }
        };
        class B : public A {
        public:
        void disp() {
        cout << "Subclass function";
        }
        };
        int main() {
        A obj;
        obj.display();
        B obj2;
        obj2.display();
        return 0;
        }
    
```

O/p :- Superclass function
Subclass function

//runtime polymorphism:-



The image shows a Visual Studio Code editor window with a C++ file named `runtime.cpp`. The code defines two classes, `A` and `B`, where `B` inherits from `A`. Both classes have a `disp()` method. Class `A`'s `disp()` prints "Super Class Function", and Class `B`'s `disp()` prints "Sub Class Function". The `main` function creates an object of `A` and an object of `B`, and calls `disp()` on both.

```
1 #include <iostream>
2 using namespace std;
3 class A {
4 public:
5     void disp(){
6         cout<<"Super Class Function"<<endl;
7     }
8 };
9 class B: public A{
10 public:
11     void disp(){
12         cout<<"Sub Class Function";
13     }
14 };
15 int main() {
16
17     A obj;
18     obj.disp();
19     B obj2;
20     obj2.disp();
21     return 0;
22 }
```

The Command Prompt window shows the execution of the program. It displays the output of the `disp()` methods for both objects, demonstrating runtime polymorphism.

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++lpu\assignment2
C:\Users\hp\Desktop\c++lpu\assignment2>g++ runtime.cpp
C:\Users\hp\Desktop\c++lpu\assignment2>a
Super Class Function
Sub Class Function
C:\Users\hp\Desktop\c++lpu\assignment2>
```

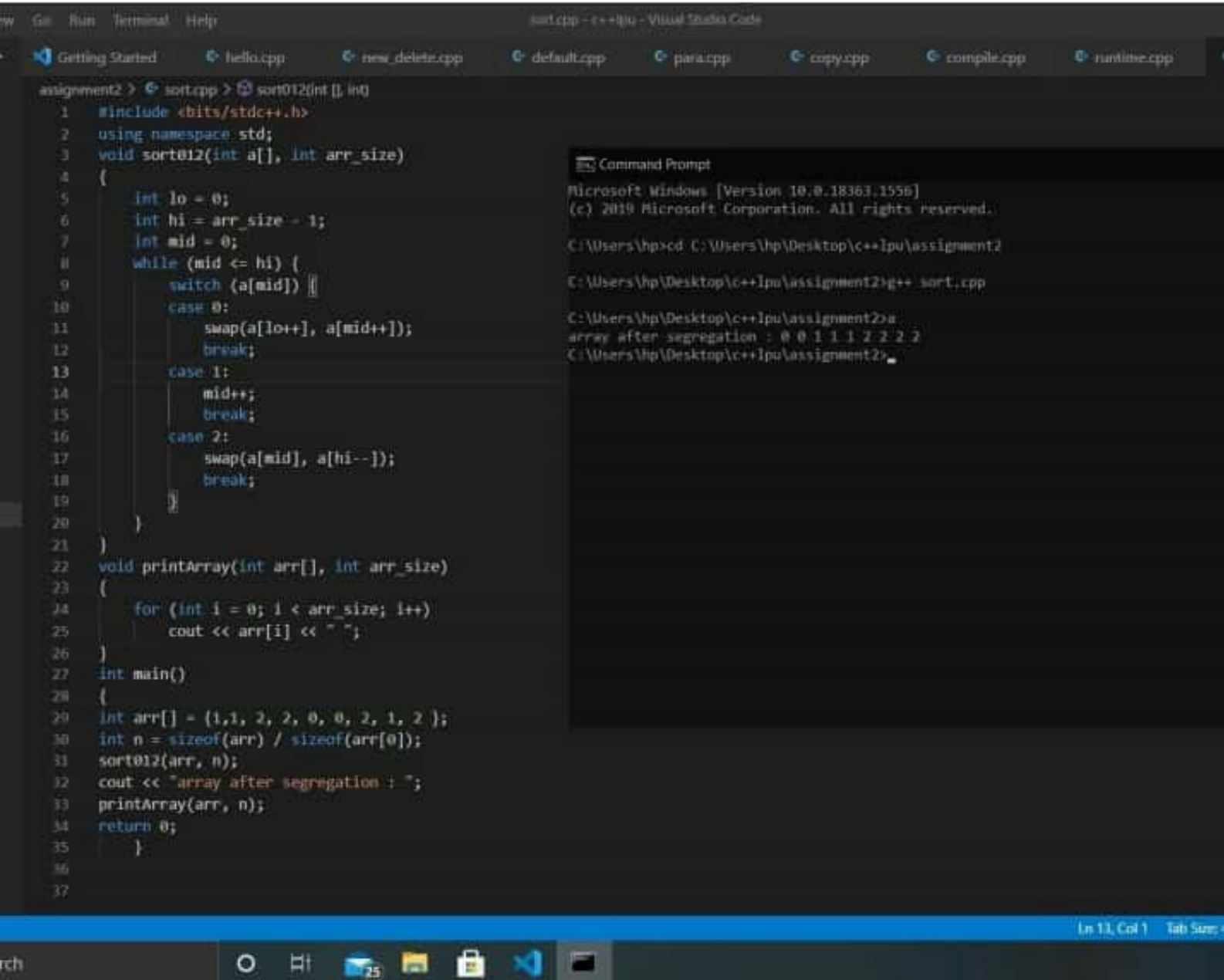

Q2) Write a program to sort an array of 0,1,2 in the best possible time and space complexity
For example :-

Input Array :- 112200212

Output Array :- 001112222

Ans → Code :- #include <bits/stdc++.h>
Using namespace std;
Void sort 012 (int a[], int arr-size)
{
int lo = 0;
int hi = arr-size-1;
int mid = 0;
while (mid <= hi) {
Switch (a[mid]) {
Case 0:
Swap (a[lo++], a[mid++]);
break;
Case 1:
mid++;
break;
Case 2:
Swap (a[mid], a[hi--]);
break;
}}}
Void Print Array (int arr[], int arr-size)
{
for (int i = 0; i < arr-size; i++)
cout << arr[i] << " ";
}

// array after segmentation:-



The image shows a Visual Studio Code editor window with a C++ file named `sort.cpp`. The code implements a partitioning function `sort012` and a `main` function. The `sort012` function uses a while loop and a switch statement to partition an array around a pivot. The `main` function initializes an array `{1, 1, 2, 2, 0, 0, 2, 1, 2}` and prints the array after segmentation. A Command Prompt window is open, showing the execution of the program and the output: `array after segregation : 0 0 1 1 1 2 2 2 2`.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 void sort012(int a[], int arr_size)
4 {
5     int lo = 0;
6     int hi = arr_size - 1;
7     int mid = 0;
8     while (mid <= hi) {
9         switch (a[mid]) {
10             case 0:
11                 swap(a[lo++], a[mid++]);
12                 break;
13             case 1:
14                 mid++;
15                 break;
16             case 2:
17                 swap(a[mid], a[hi--]);
18                 break;
19         }
20     }
21 }
22 void printArray(int arr[], int arr_size)
23 {
24     for (int i = 0; i < arr_size; i++)
25         cout << arr[i] << " ";
26 }
27 int main()
28 {
29     int arr[] = {1, 1, 2, 2, 0, 0, 2, 1, 2};
30     int n = sizeof(arr) / sizeof(arr[0]);
31     sort012(arr, n);
32     cout << "array after segregation : ";
33     printArray(arr, n);
34     return 0;
35 }
```

Command Prompt

Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\hp>cd C:\Users\hp\Desktop\c++Ipu\assignment2

C:\Users\hp\Desktop\c++Ipu\assignment2>g++ sort.cpp

C:\Users\hp\Desktop\c++Ipu\assignment2>a

array after segregation : 0 0 1 1 1 2 2 2 2

C:\Users\hp\Desktop\c++Ipu\assignment2>


```

int main()
{
    int arr[] = {1, 1, 2, 2, 0, 0, 2, 1, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    Sort 0/2 (arr, n);
    cout << "array after segregation:";
    PrintArray(arr, n);
    return 0;
}

```

o/p → array after segregation: 0 0 1 1 2 2 2

Q3) Create a class named 'Member' having the following members:

Data members

1- Name

4- Address

2- Age

5- Salary

3- Phone number

It also has a method named 'Print Salary' which prints the salary of members. Two classes 'Employee' and 'manager' inherit the 'member' class. The 'Employee' and 'manager' classes have data members 'specialization' and 'department' respectively. Now assign name, age, Phone no., address and salary to an employee and a manager by making an object of both these classes and print the same.

Ans → Code :- `#include <bits/stdc++.h>`
`Using namespace std;`

```

Class member {
    char name [40], address [50];
    int address number;
    int age;
    Public:
    int salary;
    void input ()
    {
        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 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 {  
    char specialization [30], department [20];  
public:  
    void input ()
```

```
{  
    cout << "\n Enter employee details : ";  
    member :: input ();  
    cout << "specialization : " << endl;  
    cin >> specialization;  
    cout << "Department : " << endl;  
    cin >> department;  
}
```

```
void display ()
```

```
{  
    cout << "\n Displaying Employee Details \n";  
    member :: display ();  
    cout << "Specialization : " << specialization << endl;  
    cout << "Department : " << department << endl;  
}
```

```
void PrintSalary ()
```

```
{  
    cout << "\n salary of the member is : " << salary << endl;  
}
```

```
class manager : public member {  
    char specialization [30], department [20];  
public :
```

```
    cout << "\n Enter manager details \n";  
    member :: input ();  
    cin >> cout << "Specialization : " << endl;  
    cin >> specialization;
```

```

    Cout << "Department : " << endl;
    Cin >> department;
}
Void display ()
{
    Cout << "\n Displaying manager details \n ";
    member :: display ();
    Cout << "Specialization : " << specialization << endl;
    Cout << "Department : " << department << endl;
}
Void Printsalary ()
{
    Cout << "\n Salary of the member is : " << salary << endl;
}
int main ()
{
    employee e;
    manager m;
    e.input ();
    e.display ();
    e.printsalary ();
    m.display ();
    m.printsalary ();
    return 0;
}

```

O/P :- Enter Employee Details :

Name : John

Age : 21

Phone Number : 0123456789

Address : Punjab

Salary : 12000000

Specialization : developer

Department : CSE

Enter Manager Details :

Name : Larvaib

Age : 28

Phone Number : 012345678

Address : Bgp

Salary : 1280000

Specialization : Coder

Department : CSE

Displaying Employee Details :

Name : John

Age : 21

Phone number : 0123456789

Address : Punjab

Salary : 12000000
Specialization : developer

Department : CSE

Salary of the member is : 12000000

Displaying Manager Details :

Name : Larvaib

Age : 28

Phone Number : 012345678

Address : Bgp

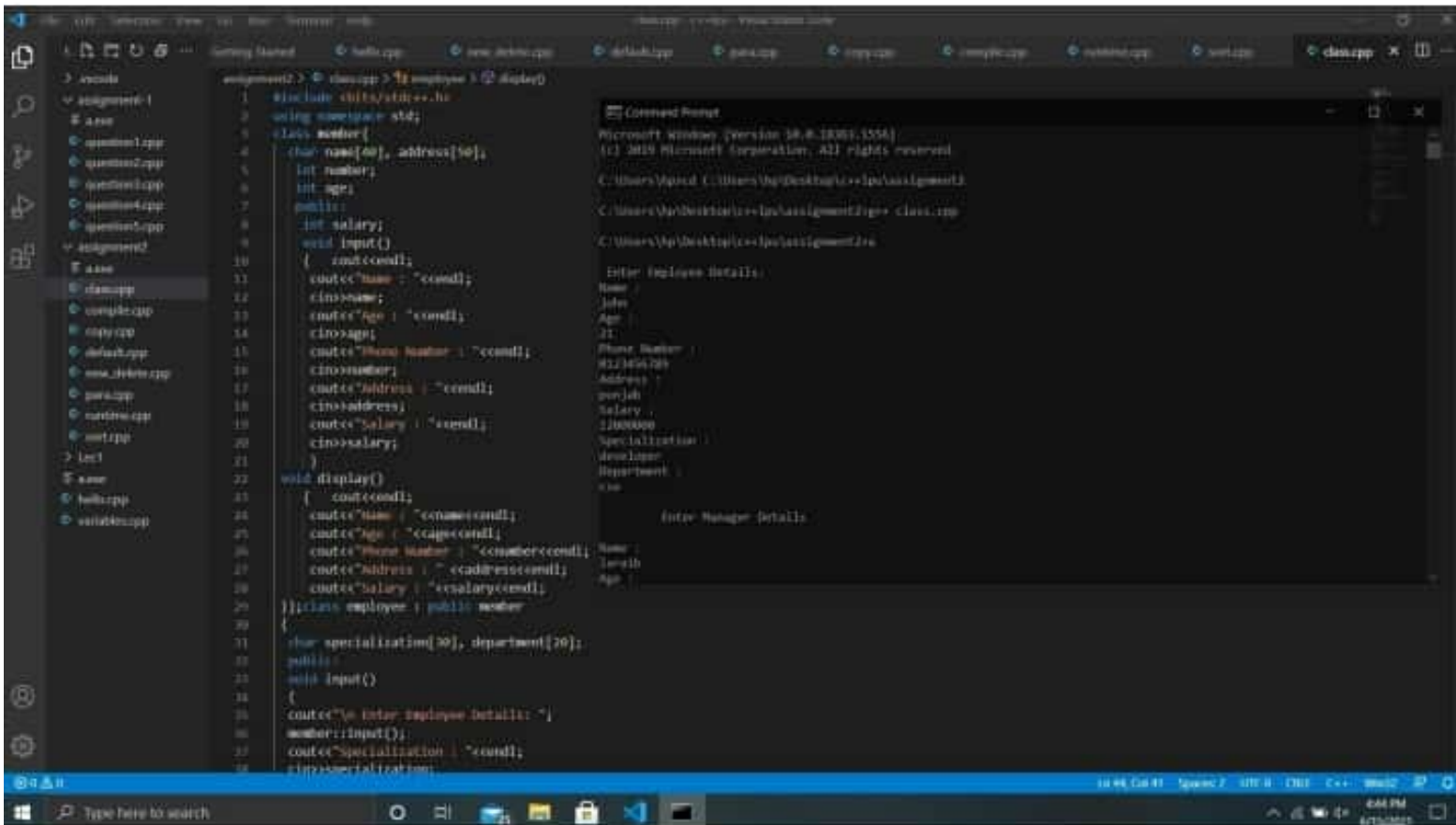
Salary : 1280000

Specialization : Coder

Department : CSE

Salary of the member is : 1280000

//inheritance:-



```
1 #include <iostream>
2 using namespace std;
3 class member{
4     char name[50], address[50];
5     int number;
6     int age;
7 public:
8     int salary;
9     void input()
10    {
11        cout<<"name : ";
12        cin>>name;
13        cout<<"age : ";
14        cin>>age;
15        cout<<"Phone Number : ";
16        cin>>number;
17        cout<<"Address : ";
18        cin>>address;
19        cout<<"Salary : ";
20        cin>>salary;
21    }
22 void display()
23 {
24     cout<<"name : " <<name<<endl;
25     cout<<"age : " <<age<<endl;
26     cout<<"Phone Number : " <<number<<endl;
27     cout<<"Address : " <<address<<endl;
28     cout<<"Salary : " <<salary<<endl;
29 }
30 class employee : public member
31 {
32     char specialization[50], department[20];
33 public:
34     void input()
35     {
36         cout<<"\n Enter Employee Details : ";
37         member::input();
38         cout<<"Specialization : ";
39         cin>>specialization;
40     }
41 };
42 class Manager : public member
43 {
44     char specialization[50], department[20];
45 public:
46     void input()
47     {
48         cout<<"\n Enter Manager Details : ";
49         member::input();
50         cout<<"Specialization : ";
51         cin>>specialization;
52         cout<<"Department : ";
53         cin>>department;
54     }
55 void display()
56 {
57     cout<<"\n Displaying Employee Details : ";
58     member::display();
59     cout<<"Specialization : " <<specialization<<endl;
60     cout<<"Department : " <<department<<endl;
61 }
62 void printSalary()
63 {
64     cout<<"\n Salary of the member is : " <<salary<<endl;
65 }
66 };
67 int main()
68 {
69     Employee e;
70     Manager m;
71     e.input();
72     m.input();
73     e.display();
74     m.display();
75     e.printSalary();
76     m.printSalary();
77 }
```

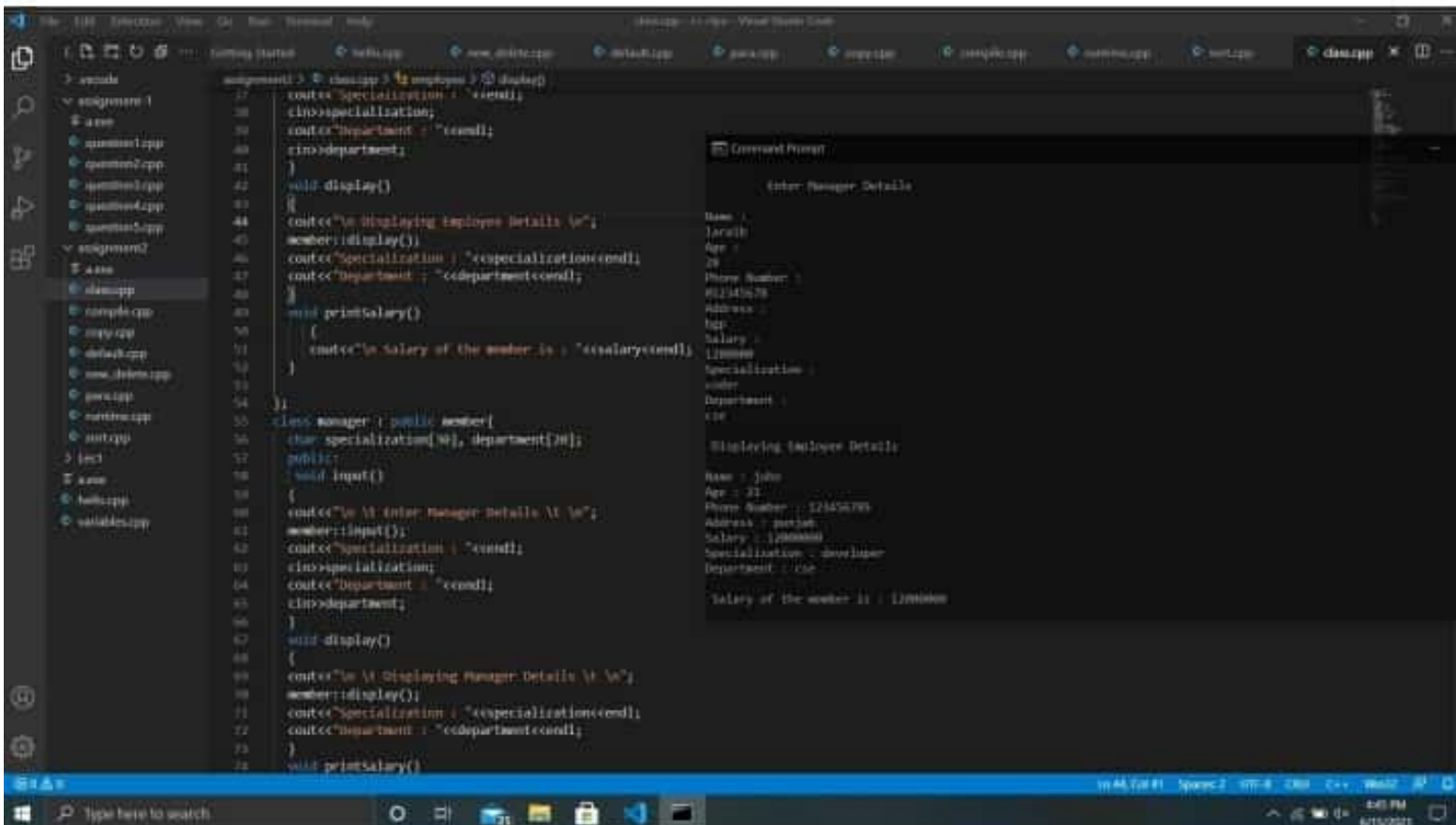
Command Prompt

Microsoft Windows [version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ajay> cd C:\Users\Ajay\Desktop\c++\assignment2
C:\Users\Ajay\Desktop\c++\assignment2> class.cpp
C:\Users\Ajay\Desktop\c++\assignment2> g++ class.cpp
C:\Users\Ajay\Desktop\c++\assignment2> .\class.exe

Enter Employee Details:

Name : John
Age : 21
Phone Number : 8123456789
Address : per job
Salary : 12000000
Specialization : developer
Department : c++



```
1 #include <iostream>
2 using namespace std;
3 class member{
4     char name[50], address[50];
5     int number;
6     int age;
7 public:
8     int salary;
9     void input()
10    {
11        cout<<"name : ";
12        cin>>name;
13        cout<<"age : ";
14        cin>>age;
15        cout<<"Phone Number : ";
16        cin>>number;
17        cout<<"Address : ";
18        cin>>address;
19        cout<<"Salary : ";
20        cin>>salary;
21    }
22 void display()
23 {
24     cout<<"name : " <<name<<endl;
25     cout<<"age : " <<age<<endl;
26     cout<<"Phone Number : " <<number<<endl;
27     cout<<"Address : " <<address<<endl;
28     cout<<"Salary : " <<salary<<endl;
29 }
30 class employee : public member
31 {
32     char specialization[50], department[20];
33 public:
34     void input()
35     {
36         cout<<"\n Enter Employee Details : ";
37         member::input();
38         cout<<"Specialization : ";
39         cin>>specialization;
40     }
41 };
42 class Manager : public member
43 {
44     char specialization[50], department[20];
45 public:
46     void input()
47     {
48         cout<<"\n Enter Manager Details : ";
49         member::input();
50         cout<<"Specialization : ";
51         cin>>specialization;
52         cout<<"Department : ";
53         cin>>department;
54     }
55 void display()
56 {
57     cout<<"\n Displaying Employee Details : ";
58     member::display();
59     cout<<"Specialization : " <<specialization<<endl;
60     cout<<"Department : " <<department<<endl;
61 }
62 void printSalary()
63 {
64     cout<<"\n Salary of the member is : " <<salary<<endl;
65 }
66 };
67 int main()
68 {
69     Employee e;
70     Manager m;
71     e.input();
72     m.input();
73     e.display();
74     m.display();
75     e.printSalary();
76     m.printSalary();
77 }
```

Command Prompt

Enter Manager Details:

Name : John
Age : 21
Phone Number : 8123456789
Address : per job
Salary : 12000000
Specialization : developer
Department : c++

Displaying Employee Details:

Name : John
Age : 21
Phone Number : 8123456789
Address : per job
Salary : 12000000
Specialization : developer
Department : c++

Salary of the member is : 12000000

