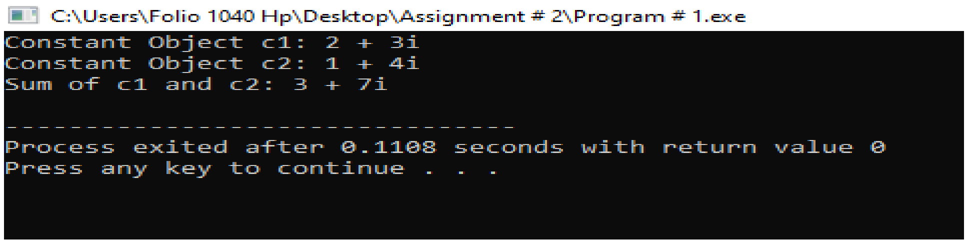
Program # 1:
input:

```
//Taibah Shahbaz
//2023-BSAI-024
//section A
//oop assignment
//Write a user-defined program to declare a class which stores a complex number. Demonstrate the use of constant objects, constant member function and constant arguments, using this class.
#include <iostream>
using namespace std;
class taibah {
private:
    double real;
    double imag;


public:
    // Constructor
    taibah(double r = 0.0, double i = 0.0)
{
        real = r;
        imag = i;
    }


    // Constant member function to display complex number
    void display() const
{
      cout << real << " + " << imag << "i" << endl;
    }


    // Constant member function to add two complex numbers
    taibah add(const taibah& other) const
{
       return taibah(real+other.real,imag+other.imag);
   }
};


int main() {
    // Constant object declaration
    const taibah t1(2.0, 3.0);
    const taibah t2(1.0, 4.0);
    // Display constant objects
    cout << "Constant Object c1: ";
    t1.display();
    cout << "Constant Object c2: ";
    t2.display();


    // Addition of constant objects
    const taibah sum = t1.add(t2);
    cout << "Sum of c1 and c2: ";
    sum.display();


    return 0;
}
```
Output:



```
C:\Users\Folio 1040 Hp\Desktop\Assignment # 2\Program # 1.exe
Constant Object c1: 2 + 3i
Constant Object c2: 1 + 4i
Sum of c1 and c2: 3 + 7i

--------------------------------
Process exited after 0.1108 seconds with return value 0
Press any key to continue . . .
```

Program # 2:

Input:

```
//Taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT

//write a class that contains the following attributes

//name of car
//direction of car
//position from imaginary zero point

//the class has following members

//the contructor to be initialized

//turn function use to change the direction of car to one steps rightside (e.g. if the direction is E
should be change to s and so on )
```

//overload the turn function to change the direction to any side directly. it should accept the direction parameter.`
//move function to change the position of car away from zero point.it should accept the distance as parameters.

```
#include <iostream>
using namespace std;

class Car {
private:
    string name;
    char direction;
    int positionX;
    int positionY;

public:
    Car(const string& name, char direction, int positionX, int positionY)
        : name(name), direction(direction), positionX(positionX), positionY(positionY) {}

    void turn() {
        switch(direction) {
            case 'N': direction = 'E'; break;
            case 'E': direction = 'S'; break;
            case 'S': direction = 'W'; break;
            case 'W': direction = 'N'; break;
            default: break;
        }
    }

    void turn(char newDirection) {
        if (newDirection == 'N' || newDirection == 'E' || newDirection == 'S' || newDirection == 'W') {
            direction = newDirection;
        }
    }

    void move(int distance) {
        switch(direction) {
            case 'N': positionY += distance; break;
            case 'E': positionX += distance; break;
            case 'S': positionY -= distance; break;
            case 'W': positionX -= distance; break;
            default: break;
        }
    }

    void display() {
        cout << "Car: " << name <<endl;
 cout<<" Position:" << positionX << ", " << positionY << endl;
cout<< " Direction: " << direction <<endl;
    }
};

int main() {
    Car taibah("Kia", 'N', 0, 0);
    taibah.display();

    taibah.turn();
    taibah.display();

    taibah.turn('W');
    taibah.display();

    taibah.move(5);
    taibah.display();

    return 0;
}
```

Output:

```
Car: Kia
Position:0, 0
Direction: N
Car: Kia
Position:0, 0
Direction: E
Car: Kia
Position:0, 0
Direction: W
Car: Kia
Position:-5, 0
Direction: W
```

Program # 3:

Input:

```
//taibah shahbaz
//2023-BSAI-024
```

```
//OOP ASSIGNMENT

//Write a function find(…) that accepts a one-dimensional integer array of size 10 as an
argument to the
//function. Your program then finds the location and value of the largest and second-largest
elements in a
//one-dimensional array.
// Display answers in main().

#include <iostream>
using namespace std;

void find(int taibah[], int size)
{
   int max = INT_MIN, secondMax = INT_MIN;
   int maxIndex = -1, secondMaxIndex = -1;

   for (int i = 0; i < size; ++i)
 {
      if (taibah[i] > max)
 {
         secondMax = max;
         secondMaxIndex = maxIndex;
         max = taibah[i];
         maxIndex = i;
      }
else if (taibah[i] > secondMax)
{
         secondMax = taibah[i];
         secondMaxIndex = i;
      }
   }

   cout << "Largest element: " << max << " at index " << maxIndex << endl;
   cout << "Second largest element: " << secondMax << " at index " << secondMaxIndex <<
endl;
}

int main() {
   int taibah[10] = {12, 46, 7, 23, 56, 85, 32, 67, 43, 120};
   find(taibah, 10);

   return 0;
}
Output:
Largest element: 120 at index 9
Second largest element: 85 at index 5
```

Program # 4:

```
//taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT

//Write a function arrange(…) that accepts a one-dimensional integer array of size 10 as an
argument to the
//function.
// The program then shifts negative numbers to the left and positive numbers to the right side of
//the array.
//For example,
//Array is
//3 -5 1 2 7 0 -15 6 -4 -8
//Output (After Deletion):
//-5 -15 -4 -8 3 1 2 7 0 6

#include <iostream>
using namespace std;

void arrange(int taibah[], int size)
{
    int left = 0, right = size - 1;

    while (left <= right)
 {
        if (taibah[left] < 0 && taibah[right] >= 0)
 {
            int temp = taibah[left];
            taiabh[left] = taibah[right];
            taibah[right] = temp;
            left++;
            right--;
        }
 else
  {
            if (taibah[left] >= 0)
 {
                left++;
            }
            if (taibah [right] < 0)
{
                right--;
            }
        }
    }
}

int main() {
    int taibah[10] = {3, -5, 0, 2, 7, 4, -15, 6, -4, -8};
    int size = 10;

    arrange(taibah, size);

    cout << "Output: ";
    for (int i = 0; i < size; ++i) {
        cout << taibah[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```
Output: 3 6 0 2 7 4 -15 -5 -4 –8
```

Program # 5:
Input:

```
//taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT

//Create a class employee which stores is name, ID and salary of an employee by user input.
The ID should
//be generated upon the creation of object, starting from 1. Include all the constructors and
destructor in
//the class. Create one object using each of the constructors and display it.

#include <iostream>
using namespace std;

class Employee
{
private:
   static int nextId;
    int id;
    std::string name;
    double salary;

public:
    Employee() {
        id = ++nextId;
        name = "jennie";
        salary = 12000;
    }

    Employee(const string& name, double salary) {
        id = ++nextId;
        this->name = name;
        this->salary = salary;
    }

    ~Employee() {
        cout << "Employee " << id << " is being deleted" << endl;
    }

    void display() {
        cout << "Employee ID: " << id << endl;
        cout << "Name: " << name << endl;
        cout << "Salary: " << salary << endl;
    }
};

int Employee::nextId = 0;

int main() {
    Employee taibah1;
    taibah1.display();

    Employee taibah2("vanessa", 50000.0);
    taibah2.display();

    return 0;
}
```

Output:

```
Employee ID: 1
```

```
Name: jennie
Salary: 12000
Employee ID: 2
Name: vanessa
Salary: 50000
Employee 2 is being deleted
Employee 1 is being deleted
```

Program # 6:
Input:

```
//taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT

//Write a C++ program for the class vehicle and its drive class water transport, road transport
and air
//transport vehicles.
//Make suitable data variables and member functions.
// When you create an object mustbe count and display total no of object created also create
every class objects and access member through
//the member functions.

#include <iostream>
using namespace std;

class Vehicle {
private:
    static int count;

public:
    Vehicle() {
        count++;
    }

    virtual ~Vehicle() {}

    virtual void display() {
        cout << "Vehicle" << endl;
    }

    static int getCount() {
        return count;
    }
};

int Vehicle::count = 0;

class WaterTransport : public Vehicle {
public:
    void display() override {
        cout << "Water Transport" << endl;
    }
};

class RoadTransport : public Vehicle {
public:
    void display() override {
        cout << "Road Transport" <<endl;
    }
};

class AirTransport : public Vehicle {
public:
    void display() override {
        cout << "Air Transport" <<endl;
    }
};

int main() {
    WaterTransport taibah1;
    RoadTransport taibah2;
    AirTransport taibah3;

    cout << "Total number of objects created: " << Vehicle::getCount() << endl;

    taibah1.display();
    taibah2.display();
    taibah3.display();

    return 0;
}
```
Output:
```
Total number of objects created: 3
Water Transport
```

Road Transport
Air Transport

Program # 7:
Input:

```
//taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT
//Implement a C++ class named Employee with the following specifications:
//The class should have private data members name (string), id (integer), and salary
(floatingpoint).
//Implement a static data member totalEmployees to keep track of the total number of
employees.
//Implement a static member function averageSalary() that calculates and returns the average
salary of all employees.
//Provide member functions to set and get the values of name, id, and salary.
//Implement a constructor to initialize the name, id, and salary of an employee.
//Implement a destructor to decrement the totalEmployees count when an object is destroyed.
#include <iostream>
#include <string>

using namespace std;

class Employee {
public:
   string name;
   int id;
   float salary;
   static int totalEmployees;
   static float totalSalary;

public:
   // Constructor
   Employee(string name, int id, float salary) : name(name), id(id), salary(salary) {
      totalEmployees++;
      totalSalary += salary;
   }

   // Destructor
   ~Employee() {
      totalEmployees--;
      totalSalary -= salary;
   }
```

```
    // Setter functions
    void setName(string name) {
        this->name = name;
    }

    void setId(int id) {
        this->id = id;
    }

    void setSalary(float salary) {
        totalSalary -= this->salary;
        this->salary = salary;
        totalSalary += this->salary;
    }

    // Getter functions
    string getName() const {
        return name;
    }

    int getId() const {
        return id;
    }

    float getSalary() const {
        return salary;
    }

    // Static member function to calculate average salary
    static float averageSalary() {
        if (totalEmployees == 0) {
            return 0.0f;
        }
        return totalSalary / totalEmployees;
    }
};

// Initialize static members
int Employee::totalEmployees = 0;
float Employee::totalSalary = 0.0f;
```

```
int main() {
    Employee t1("Taibah Shahbaz", 1, 120000.0);
    Employee t2("Tahreem butt", 2, 250000.0);

    cout << "Total employees: " << Employee::totalEmployees << endl;
    cout << "Average salary: " << Employee::averageSalary() << endl;

    return 0;
}
```

Output:

```
Total employees: 2
Average salary: 185000
```

Program # 8:

Input:

```
//taibah shahbaz
//2023-BSAI-024
//OOP ASSIGNMENT
//(Car Pool Savings Calculator) Research several car-pooling websites. create an application
that calculates your daily driving cost,
//so that you can estimate how much money could be saved by carpooling, which also has other
advantages such as reducing carbon emission
//and reducing traffic congestion. The application should input the following and display the
user's cost per day of driving to word:
//a) Total miles driven per day.
//b) Cost per gallon of gasoline.
//c) Average miles per gallon
//d) Parking fees per day.
//e) Toll per day
#include <iostream>
using namespace std;

double calculateDrivingCost(double milesPerDay, double costPerGallon, double
milesPerGallon, double parkingFee, double toll, int numPeople) {
    double gasCost = (milesPerDay / milesPerGallon) * costPerGallon;
    double totalCost = gasCost + parkingFee + toll;
    double costPerPerson = totalCost / numPeople;
    return costPerPerson;
}

int main() {
    double milesPerDay, costPerGallon, milesPerGallon, parkingFee, toll;
    int numPeople;
    cout << "\t\tCar Pool Savings Calculator\n\n";

    cout << "Enter total miles driven per day: ";
    cin >> milesPerDay;

    cout << "Enter cost per gallon of gasoline: ";
    cin >> costPerGallon;

    cout << "Enter average miles per gallon: ";
    cin >> milesPerGallon;

    cout << "Enter parking fees per day: ";
    cin >> parkingFee;

    cout << "Enter toll per day: ";
    cin >> toll;

    cout << "Enter number of people in the carpool (including yourself): ";
    cin >> numPeople;

    double dailyCostPerPerson = calculateDrivingCost(milesPerDay, costPerGallon,
milesPerGallon, parkingFee, toll, numPeople);

    cout << "\nYour daily driving cost per person: $" << dailyCostPerPerson << "\n";

    double totalSavings = (milesPerDay / milesPerGallon) * costPerGallon - dailyCostPerPerson;

    if (totalSavings > 0) {
        cout << "Money saved by carpooling per day: $" << totalSavings << "\n";
    } else {
        cout << "Carpooling does not lead to savings compared to driving alone.\n";
    }

    return 0;
}
```
Output:
```
                Car Pool Savings Calculator

Enter total miles driven per day: 323
Enter cost per gallon of gasoline: 5000
Enter average miles per gallon: 23
Enter parking fees per day: 350
Enter toll per day: 450
Enter number of people in the carpool (including yourself): 4

Your daily driving cost per person: $17754.3
Money saved by carpooling per day: $52463
```