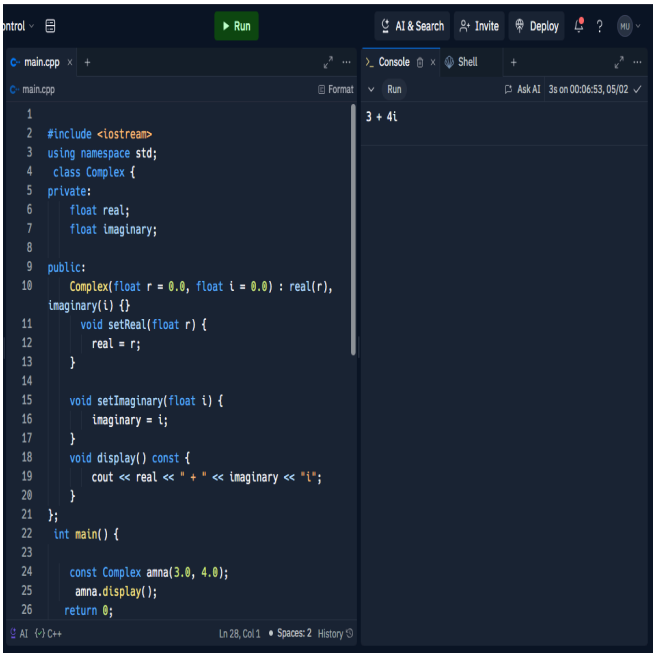
ASSIGNMENT NO: 01
REGISTRATION NUMBER: BS-AI-027
SUBMITTED TO: PROF JAVED ABBAS
SUBMITTED BY: AMNA SHAHZAD
DEPARTMENT: COMPUTER SCIENCE

Problem no 1:

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

Solution:

```
//File Name: program 1
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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 Complex {
private:
   float real;
   float imaginary;

public:
  Complex(float r = 0.0, float i = 0.0) : real(r), imaginary(i) {}
     void setReal(float r) {
       real = r;
   }

   void setImaginary(float i) {
       imaginary = i;
   }
   void display() const {
       cout << real << " + " << imaginary << "i";
   }
};
 int main() {

   const Complex amna(3.0, 4.0);
    amna.display();
  return 0;
}
```

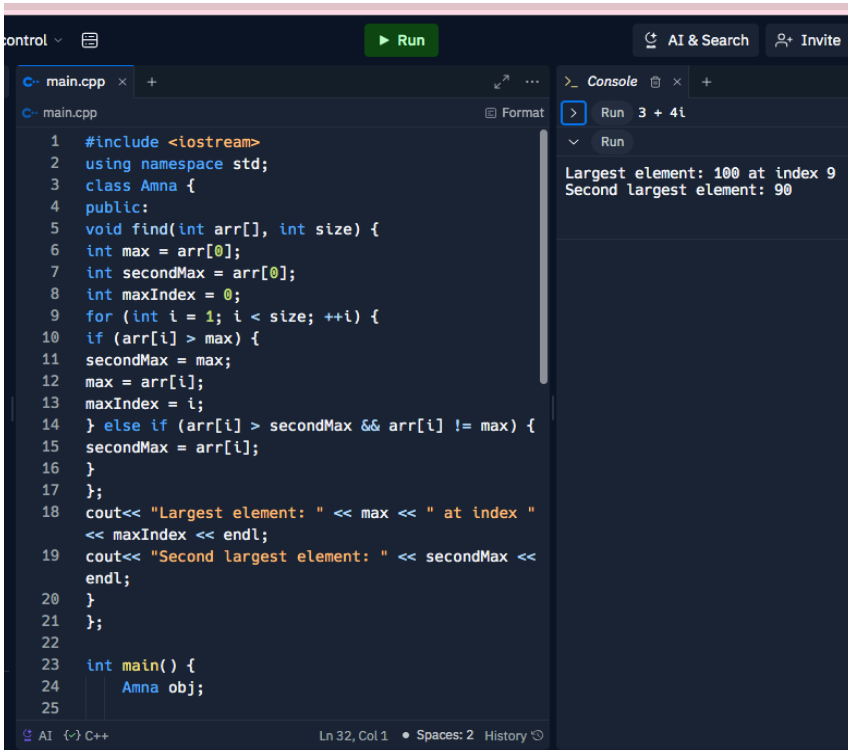


Problem no 02:

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().

Solution:

```
//File Name: program 3
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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;
class Amna {
public:
void find(int arr[], int size) {
int max = arr[0];
int secondMax = arr[0];
int maxIndex = 0;
for (int i = 1; i < size; ++i) {
if (arr[i] > max) {
secondMax = max;
max = arr[i];
maxIndex = i;
} else if (arr[i] > secondMax && arr[i] != max) {
secondMax = arr[i];
}
};
cout<< "Largest element: " << max << " at index " << maxIndex << endl;
cout<< "Second largest element: " << secondMax << endl;
}
};

int main() {
    Amna obj;

    int arr[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

    obj.find(arr, 10);

    return 0;
}
```



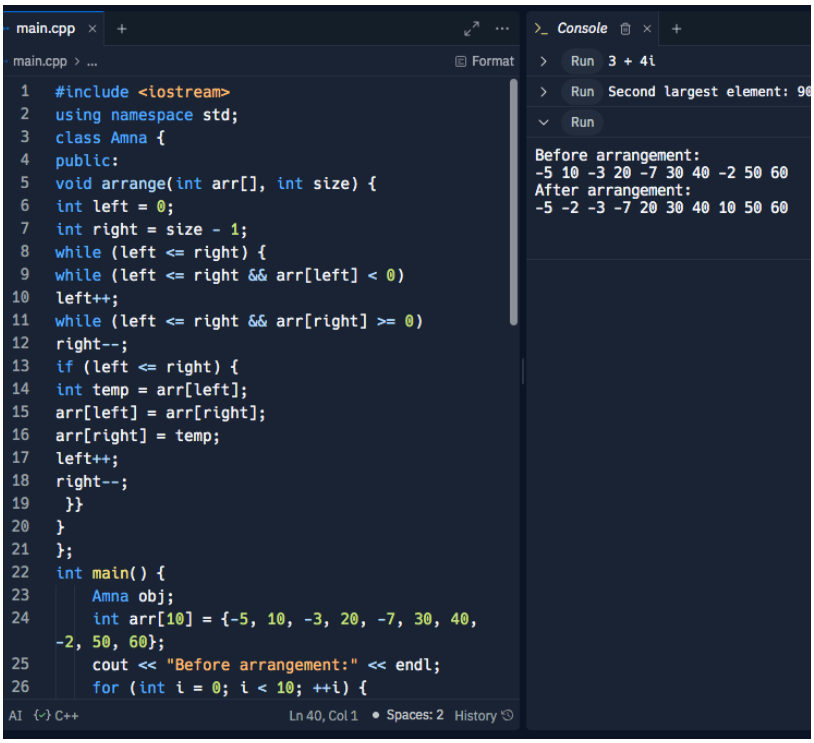
Problem no 03:

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.

Solution:

```
//File Name: program 4
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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.

#include <iostream>
using namespace std;
class Amna {
public:
void arrange(int arr[], int size) {
int left = 0;
int right = size - 1;
while (left <= right) {
while (left <= right && arr[left] < 0)
left++;
while (left <= right && arr[right] >= 0)
right--;
if (left <= right) {
int temp = arr[left];
```

```
arr[left] = arr[right];
arr[right] = temp;
left++;
right--;
 }}
}
};
int main() {
    Amna obj;
    int arr[10] = {-5, 10, -3, 20, -7, 30, 40, -2, 50, 60};
    cout << "Before arrangement:" << endl;
    for (int i = 0; i < 10; ++i) {
        cout << arr[i] << " ";
    }
   cout << endl;

    obj.arrange(arr, 10);

   cout << "After arrangement:" << endl;
    for (int i = 0; i < 10; ++i) {
     cout << arr[i] << " ";
    }
   cout << endl;
    return 0;
}
```



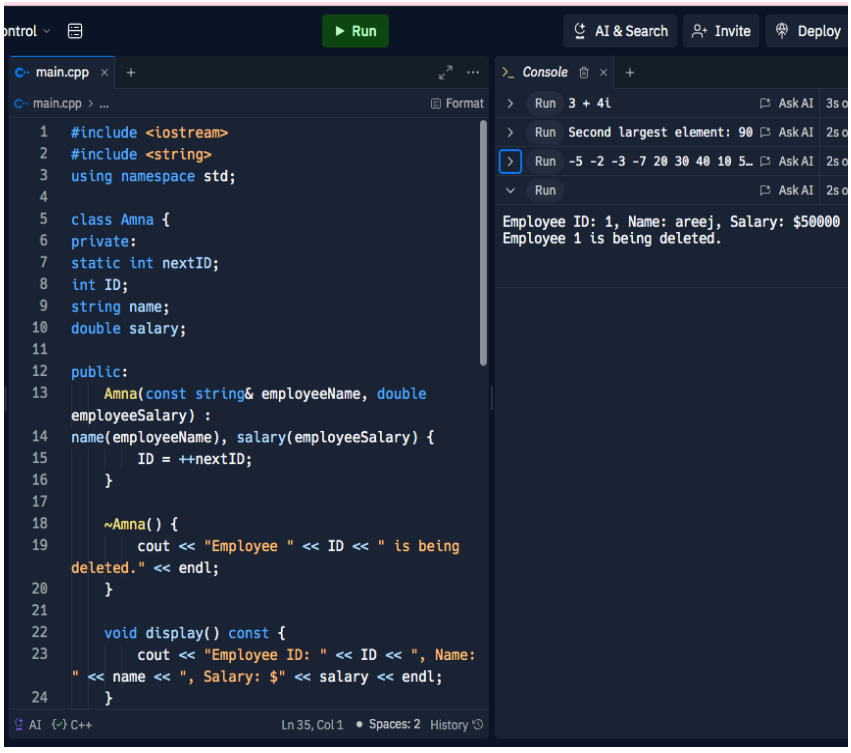
Problem no 4:

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.

Solution:

//File Name: program 5
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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>
#include <string>
using namespace std;

class Amna {
private:
static int nextID;
int ID;
string name;
double salary;

public:
   Amna(const string& employeeName, double employeeSalary) :
name(employeeName), salary(employeeSalary) {
      ID = ++nextID;
   }

   ~Amna() {
      cout << "Employee " << ID << " is being deleted." << endl;
   }

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

int Amna::nextID = 0;

int main() {
   Amna emp1("areej", 50000.0);
   emp1.display() ;

   return 0;
}
```

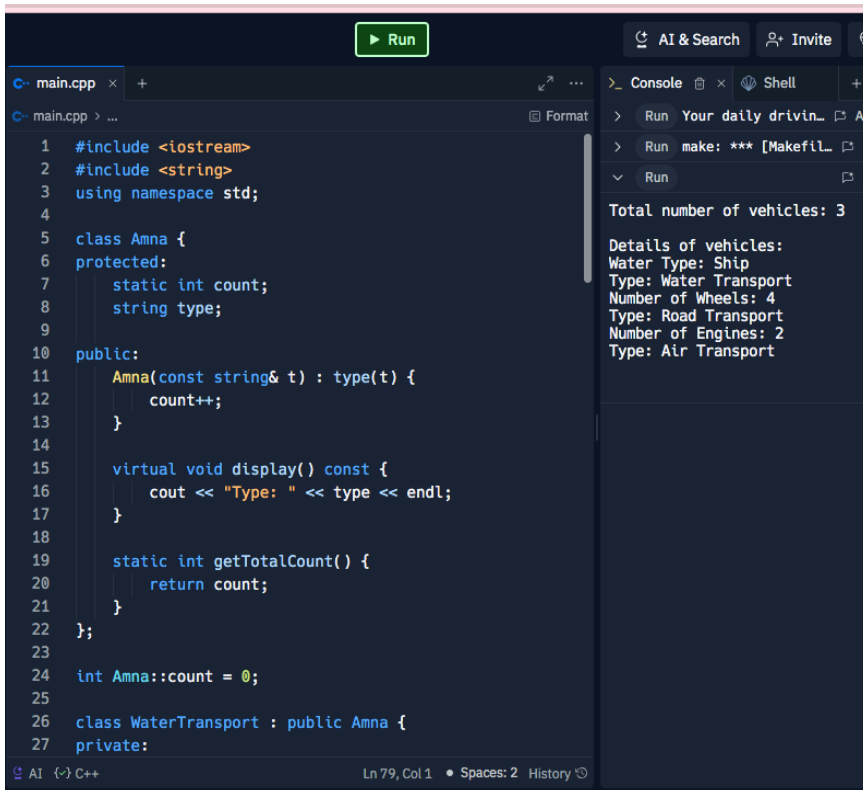


Problem no 5:

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 must be count and display total no of object created also create every class objects and access member through the member functions.

SOLUTION:

```
//File Name: program 6
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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 must be count and display total no of object created also create every class objects and
access member through the member functions.
#include <iostream>
#include <string>
using namespace std;

class Amna {
protected:
    static int count;
    string type;

public:
    Amna(const string& t) : type(t) {
        count++;
    }

    virtual void display() const {
        cout << "Type: " << type << endl;
    }

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

int Amna::count = 0;

class WaterTransport : public Amna {
private:
    string waterType;

public:
    WaterTransport(const string& wt) : Amna("Water Transport"), waterType(wt) {}
```



```
    void display() const override {
        cout << "Water Type: " << waterType << endl;
        Amna::display();
    }
};

class RoadTransport : public Amna {
private:
    int wheels;

public:
    RoadTransport(int w) : Amna("Road Transport"), wheels(w) {}

    void display() const override {
        cout << "Number of Wheels: " << wheels << endl;
        Amna::display();
    }
};

class AirTransport : public Amna {
private:
    int engines;

public:
    AirTransport(int e) : Amna("Air Transport"), engines(e) {}

    void display() const override {
        cout << "Number of Engines: " << engines << endl;
        Amna::display();
    }
};

int main() {
    WaterTransport boat("Ship");
    RoadTransport car(4);
    AirTransport airplane(2);

    cout << "Total number of vehicles: " << Amna::getTotalCount() << endl;

    cout << "\nDetails of vehicles:" << endl;
    boat.display();
    car.display();
    airplane.display();

    return 0;
}
```

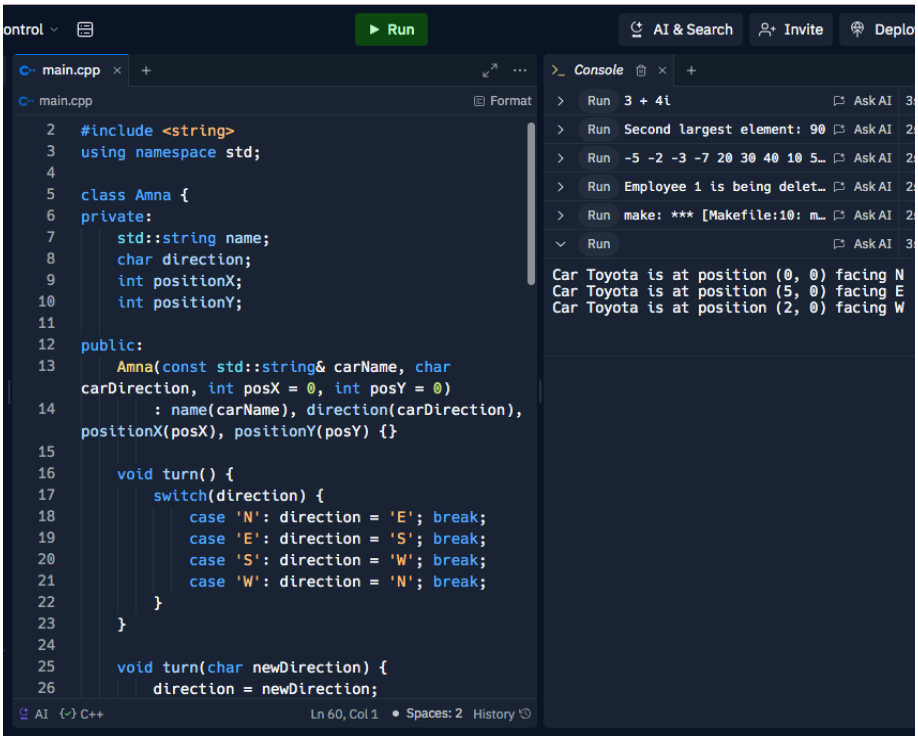
Problem no 6:

Write a class that contain the following attribute • The name of car • Direction of car (E, W, N, S) • The position of car (from imaginary zero point) The class has fallowing member function The constructor to be initialize • Turn function use to change the direction of car to one steps right side (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 parameter.

Solution:

```
//File Name: program 2
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//Write a class that contain the following attribute
//The name of car
//Direction of car (E, W, N, S)
//The position of car (from imaginary zero point)
//The class has fallowing member function
//The constructor to be initialize
 //Turn function use to change the direction of car to one steps right side (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 parameter.

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

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

public:
    Amna(const std::string& carName, char carDirection, int posX = 0, int posY = 0)
        : name(carName), direction(carDirection), positionX(posX), positionY(posY) {}

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

```
 void turn(char newDirection) {
        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;
        }
    }
 void displayPosition() const {
    cout << "Car " << name << " is at position (" << positionX << ", " << positionY << ") facing " <<
direction << endl;
    }
};

int main() {
    Amna myCar("Toyota", 'N');

    myCar.displayPosition();

    myCar.turn();
    myCar.move(5);

    myCar.displayPosition();

    myCar.turn('W');
    myCar.move(3);

    myCar.displayPosition();

    return 0;
}
```



Problem no 7:

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.

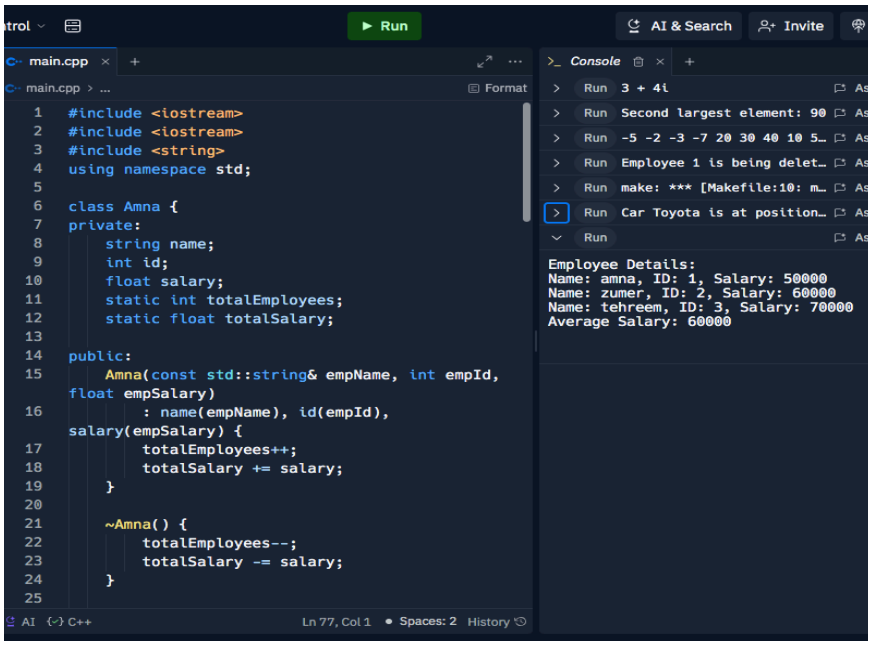
Solution:

```
//File Name: program 7
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//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 Amna {
private:
    string name;
    int id;
    float salary;
    static int totalEmployees;
    static float totalSalary;

public:
    Amna(const std::string& empName, int empId, float empSalary)
        : name(empName), id(empId), salary(empSalary) {        totalEmployees++;
        totalSalary += salary;
    }

    ~Amna() {
        totalEmployees--;
        totalSalary -= salary;
    }

    static float averageSalary() {
        if (totalEmployees == 0) {
            return 0.0;
        }
        return totalSalary / totalEmployees;
    }

    void setName(const std::string& empName) {
        name = empName;
    }

    std::string getName() const {
        return name;
    }

    void setId(int empId) {
        id = empId;
    }

    int getId() const {
        return id;
```

```
    }

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

    float getSalary() const {
        return salary;
    }
};

int Amna::totalEmployees = 0;
float Amna::totalSalary = 0.0;

int main() {

    Amna emp1("amna", 1, 50000.0);
    Amna emp2("zumer", 2, 60000.0);
    Amna emp3("tehreem", 3, 70000.0);
cout << "Employee Details:\n";
  cout << "Name: " << emp1.getName() << ", ID: " << emp1.getId() << ", Salary: " <<
emp1.getSalary() << endl;
cout << "Name: " << emp2.getName() << ", ID: " << emp2.getId() << ", Salary: " <<
emp2.getSalary() << endl;
cout << "Name: " << emp3.getName() << ", ID: " << emp3.getId() << ", Salary: " <<
emp3.getSalary() << endl;

  cout << "Average Salary: " << Amna::averageSalary() << endl;

    return 0;
}
```



Problem no 8:

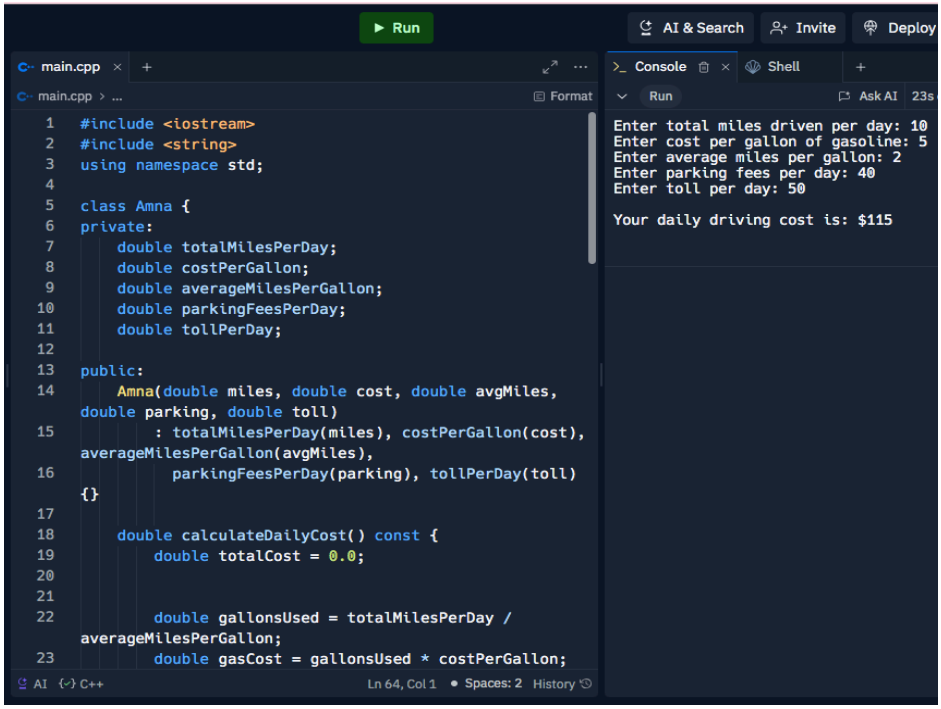
(Car Pool Savings Calculator) Research several car-pooling websites. create an application that calculates your daily driving cost, so that you can estimate how much money could be saved by carpooling, which also has other advantages such as reducing carbon emission and reducing traffic congestion. The 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

Solution:

//File Name: program 8
//Date:01-05-2024
//Name:Amna Shahzad
//Registration number:bs-ai-027
//(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>
#include <string>
using namespace std;

class Amna {
private:
   double totalMilesPerDay;
   double costPerGallon;
   double averageMilesPerGallon;
   double parkingFeesPerDay;
   double tollPerDay;

public:
   Amna(double miles, double cost, double avgMiles, double parking, double toll)
      : totalMilesPerDay(miles), costPerGallon(cost), averageMilesPerGallon(avgMiles),
        parkingFeesPerDay(parking), tollPerDay(toll) {}

   double calculateDailyCost() const {
      double totalCost = 0.0;


      double gallonsUsed = totalMilesPerDay / averageMilesPerGallon;
      double gasCost = gallonsUsed * costPerGallon;
      totalCost += gasCost;


      totalCost += parkingFeesPerDay;
      totalCost += tollPerDay;

      return totalCost;
   }
};

int main() {
   double totalMiles, costPerGallon, averageMilesPerGallon, parkingFees, toll;


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

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

   cout << "Enter average miles per gallon: ";
```

```
    cin >> averageMilesPerGallon;

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

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


    Amna calculator(totalMiles, costPerGallon, averageMilesPerGallon, parkingFees, toll);


    double dailyCost = calculator.calculateDailyCost();


    std::cout << "\nYour daily driving cost is: $" << dailyCost << endl;

    return 0;
}
```
