Q37.Create a structure STUDENT with fields ROLLNO, NAME, and MARKS. Display the student with the highest marks out of 'n' records.

Objective:

Identify and display the student with the highest marks from input records.

Description:
Use a STUDENT structure to store roll number, name, and marks. Track the highest scorer dynamically during input no array storage needed.

Code:

```
#include <iostream>
#include<string.h>
using namespace std;
struct STUDENT {
    int ROLLNO;
    string NAME;
    float MARKS;
};

int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;

    STUDENT students[n];

    for (int i = 0; i < n; i++) {
        cout << "\nEnter details for student " << i + 1 << ":\n";
        cout << "Roll No: ";
        cin >> students[i].ROLLNO;
        cin.ignore();
        cout << "Name: ";
        getline(cin, students[i].NAME);
        cout << "Marks: ";
        cin >> students[i].MARKS;
```

```
    }
    int topIndex = 0;
    for (int i = 1; i < n; i++) {
        if (students[i].MARKS > students[topIndex].MARKS) {
            topIndex = i;
        }
    }
    cout << "\nStudent with the highest marks:\n";
    cout << "Roll No: " << students[topIndex].ROLLNO << endl;
    cout << "Name   : " << students[topIndex].NAME << endl;
    cout << "Marks  : " << students[topIndex].MARKS << endl;

    return 0;
}
```

Output:

```
Enter number of students: 2
Enter details for student 1:
Roll No: 1
Name: John
Marks: 85.5

Enter details for student 2:
Roll No: 2
Name: Sarah
Marks: 91.0

Student with the highest marks:
Roll No: 2
Name   : Sarah
Marks  : 91
```

Q38.Print the address and data for an integer, float number, and character using pointers.

Determine the size of each data type.

Objective

To use pointers to print the address and value of an int, float, and char, and to determine the size of each data type.

Description: The program declares an int, float, and char variable. It uses pointers to access and display each variable's address and value. It also uses the sizeof() operator to print the size of each data type in bytes.

Code:

```
#include <iostream>

using namespace std;

int main() {
   int i = 10;
   float f = 3.14;
   char c = 'A';
   int *iptr = &i;
   float *fptr = &f;
   char *cptr = &c;
   cout << "Integer value: " << *iptr << ", Address: " << iptr << endl;
   cout << "Float value  : " << *fptr << ", Address: " << fptr << endl;
   cout << "Char value   : " << *cptr << ", Address: " << (void*)cptr << endl;


   cout << "\nSize of int  : " << sizeof(int) << " bytes" << endl;
   cout << "Size of float: " << sizeof(float) << " bytes" << endl;
   cout << "Size of char : " << sizeof(char) << " bytes" << endl;
   return 0;
}
```

Output:

```
Integer value: 10, Address: 0x61ff08
Float value  : 3.14, Address: 0x61ff04
Char value   : A, Address: 0x61ff03
Size of int  : 4 bytes
Size of float: 4 bytes
Size of char : 1 bytes
```

Q39.Write a C++ program using functions to

Find the largest of three numbers

And for counting characters, white and spaces, and digits in a given string.

Objective

1. To find the largest of three numbers using a function.
2. To count characters, white spaces, and digits in a given string using a function.

Description

The program contains two functions:

- findLargest() determines the largest among three numbers.
- analyzeString() counts letters, white spaces, and digits in a string by traversing each character.

Code:

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

int findLargest(int a, int b, int c) {
   if (a >= b && a >= c)
      return a;
   else if (b >= a && b >= c)
      return b;
   else
      return c;
}
void analyzeString(const string& str) {
   int letters = 0, spaces = 0, digits = 0;

   for (char ch : str) {
      if (isalpha(ch))
         letters++;
      else if (isdigit(ch))
         digits++;
      else if (isspace(ch))
```

```
            spaces++;
    }
    cout << "\nCharacter Analysis:" << endl;
    cout << "Letters     : " << letters << endl;
    cout << "Digits      : " << digits << endl;
    cout << "White spaces: " << spaces << endl;
}
int main() {
    int x, y, z;
    cout << "Enter three numbers: ";
    cin >> x >> y >> z;

    int largest = findLargest(x, y, z);
    cout << "Largest number: " << largest << endl;
    cin.ignore();
    string input;
    cout << "\nEnter a string: ";
    getline(cin, input);

    analyzeString(input);
    return 0;
}
```

Output:

```
Enter three numbers: 15 78 34
Largest number: 78

Enter a string: Hello World 123

Character Analysis:
Letters     : 10
Digits      : 3
White spaces: 2
```

Q40. EMPLOYEE class contains the following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary. Write member function read () and print () to enter the data and display the employee records.

Objective:

To define an EMPLOYEE class with data members for employee number, name, basic salary, DA, IT, and net salary. The program uses member functions read() to input and print() to display employee details.

Description:

The program creates an EMPLOYEE class with methods:

- read() — to accept employee details and calculate DA, IT, and Net Salary.
- print() — to display all employee information in a formatted way.

Code:

```
#include <iostream>

using namespace std;

class EMPLOYEE {

private:

   int empNo;

   string empName;

   float basic, DA, IT, netSalary;


public:

   void read() {

      cout << "Enter Employee Number: ";

      cin >> empNo;

      cin.ignore();

      cout << "Enter Employee Name: ";

      getline(cin, empName);

      cout << "Enter Basic Salary: ";

      cin >> basic;


      DA = 0.8 * basic;

      IT = 0.1 * (basic + DA);

      netSalary = basic + DA - IT;

   }
```

```
    void print() {
        cout << "\nEmployee Details:\n";
        cout << "Employee Number : " << empNo << endl;
        cout << "Employee Name   : " << empName << endl;
        cout << "Basic Salary    : " << basic << endl;
        cout << "Dearness Allow. : " << DA << endl;
        cout << "Income Tax      : " << IT << endl;
        cout << "Net Salary      : " << netSalary << endl;
    }
};
int main() {
    EMPLOYEE emp;

    emp.read();
    emp.print();

    return 0;
}
```

Output:

```
Enter Employee Number: 101
Enter Employee Name: Ravi Kumar
Enter Basic Salary: 30000

Employee Details:
Employee Number : 101
Employee Name   : Ravi Kumar
Basic Salary    : 30000
Dearness Allow. : 24000
Income Tax      : 5400
Net Salary      : 48600
```

Q41.Create a class TEACHER with data members' salary and experience. Implement the following: a) Initialize using parametrized constructors

b) Illustrate the use of the default constructor

Objective:

Create a TEACHER class to demonstrate the use of **default** and **parameterized constructors**.

Description:

Defines a TEACHER class with default and parameterized constructors to initialize salary and experience, and displays these values.

Code:

```
#include <iostream>

using namespace std;


class TEACHER {
   float salary;
   int experience;


public:
   TEACHER() {
      salary = 0;
      experience = 0;
   }


   TEACHER(float s, int e) {
      salary = s;
      experience = e;
   }
   void display() {
      cout << "Salary: " << salary << ", Experience: " << experience << " years" << endl;
   }
};
int main() {
   TEACHER t1;
   TEACHER t2(40000, 3);
```

```
    cout << "Default Constructor Output:\n";
    t1.display();

    cout << "Parameterized Constructor Output:\n";
    t2.display();

    return 0;
}
```

Output:

Default Constructor Output:

Salary: 0, Experience: 0 years

Parameterized Constructor Output:

Salary: 40000, Experience: 3 years

Q41.Create a class Employee with data members: empID, name. Include a method to input and display this information.Now, derive a class Manager from Employee that adds an additional data member department.Implement input and display functions in both classes.

In main(), create an object of Manager and demonstrate the functionality.

Objective:

Create an Employee class and a derived Manager class with input and display functions. Demonstrate with a Manager object.

Description:

The program defines an Employee class with ID and name, and a Manager class that inherits Employee and adds a department. Both classes have input and display methods. The Manager object uses these to show employee details including department.

Code:

```
#include <iostream>

using namespace std;


class Employee {

protected:

   int empID;

   string name;


public:

   void input() {

      cout << "Employee ID: ";

      cin >> empID;

      cin.ignore();

      cout << "Name: ";

      getline(cin, name);

   }


   void display() {

      cout << "ID: " << empID << ", Name: " << name << endl;

   }

};
```

```
class Manager : public Employee {
    string department;

public:
    void input() {
        Employee::input();
        cout << "Department: ";
        getline(cin, department);
    }

    void display() {
        Employee::display();
        cout << "Department: " << department << endl;
    }
};
int main() {
    Manager mgr;
    mgr.input();
    mgr.display();
    return 0;
}
```

Output:

```
Employee ID: 101
Name: John Doe
Department: Marketing
ID: 101, Name: John Doe
Department: Marketing
```

Q42.Create a class Academics with data members: marks[5].Create another class Sports with a data member: sportsScore.Derive a class Result from both Academics and Sports.Add a method to calculate the average of academic marks and total score including sports.Display all the details in the output.

Objective:

Create classes Academics and Sports, derive Result from both, calculate average marks and total score including sports, and display all details.

Description:

- Academics stores marks of 5 subjects.
- Sports stores a sports score.
- Result inherits from both, calculates the average of marks and total score (average + sports score), and displays all details.

Code:

```
#include <iostream>

using namespace std;


class Academics {

protected:

   int marks[5];


public:

   void inputMarks() {

      cout << "Enter marks for 5 subjects: ";

      for (int i = 0; i < 5; i++) {

         cin >> marks[i];

      }

   }

};


class Sports {

protected:

   int sportsScore;
```

```
public:
    void inputSportsScore() {
        cout << "Enter sports score: ";
        cin >> sportsScore;
    }
};

class Result : public Academics, public Sports {
    float average;
    int totalScore;

public:
    void calculate() {
        int sum = 0;
        for (int i = 0; i < 5; i++) {
            sum += marks[i];
        }
        average = sum / 5.0;
        totalScore = sum + sportsScore;
    }

    void display() {
        cout << "\nMarks: ";
        for (int i = 0; i < 5; i++) {
            cout << marks[i] << " ";
        }
        cout << "\nSports Score: " << sportsScore;
        cout << "\nAverage Marks: " << average;
        cout << "\nTotal Score (Academics + Sports): " << totalScore << endl;
    }
};
```

```
int main() {
    Result r;
    r.inputMarks();
    r.inputSportsScore();
    r.calculate();
    r.display();
    return 0;
}
```

Output:

Enter marks for 5 subjects: 80 75 90 85 70

Enter sports score: 40

Marks: 80 75 90 85 70

Sports Score: 40

Average Marks: 80

Total Score (Academics + Sports): 380

Q43. .Design a base class Person that contains name and age.Derive a class Student from Person that adds rollNumber and class.Further derive a class Graduate from Student that includes degree, university, and year.Create methods in each class to accept and display respective data.Demonstrate the multilevel inheritance and data flow using appropriate object creation.

Objective:

Demonstrate multilevel inheritance using classes: Person → Student → Graduate with data input and display methods.

Description:

- Person class contains name and age.
- Student is derived from Person and adds rollNumber and class.
- Graduate is derived from Student and adds degree, university, and year.
- Each class has methods to input and display data.
- A Graduate object demonstrates data flow across all levels.

Code:

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

class Person {
protected:
    string name;
    int age;

public:
    void inputPerson() {
        cout << "Enter name: ";
        getline(cin, name);
        cout << "Enter age: ";
        cin >> age;
        cin.ignore();
    }
    void displayPerson() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};
```

```
class Student : public Person {
protected:
    int rollNumber;
    string studentClass;
public:
    void inputStudent() {
        inputPerson();
        cout << "Enter roll number: ";
        cin >> rollNumber;
        cin.ignore();
        cout << "Enter class: ";
        getline(cin, studentClass);
    }
    void displayStudent() {
        displayPerson();
        cout << "Roll No: " << rollNumber << ", Class: " << studentClass << endl;
    }
};
class Graduate : public Student {
    string degree, university;
    int year;
public:
    void inputGraduate() {
        inputStudent();
        cout << "Enter degree: ";
        getline(cin, degree);
        cout << "Enter university: ";
        getline(cin, university);
        cout << "Enter passing year: ";
        cin >> year;
    }
    void displayGraduate() {
```

```
        displayStudent();
        cout << "Degree: " << degree << ", University: " << university
             << ", Year: " << year << endl;
    }
};
int main() {
    Graduate g;
    cout << "Enter Graduate Details:\n";
    g.inputGraduate();

    cout << "\nGraduate Information:\n";
    g.displayGraduate();

    return 0;
}
```

Output:

```
Enter Graduate Details:
Enter name: Priya Sharma
Enter age: 22
Enter roll number: 102
Enter class: BCA Final
Enter degree: BCA
Enter university: GEHU
Enter passing year: 2024

Graduate Information:
Name: Priya Sharma, Age: 22
Roll No: 102, Class: BCA Final
Degree: BCA, University: GEHU Year 2025
```

Q44.Create a base class Shape with a method getData() and a virtual method area().Derive two classes:Circle with radius and Rectangle with length and breadth.Override the area() method in each derived class to compute respective areas.In main(), create objects of both Circle and Rectangle, input dimensions, and display their areas.

Objective:

Use virtual functions in a base class Shape, and override area() in derived classes Circle and Rectangle.

Description:

- Shape is a base class with a getData() method and a **virtual** area() method.
- Circle and Rectangle inherit Shape and override area() to calculate their areas.
- In main(), objects of both are created, data is input, and areas are displayed using polymorphism.

Code:

```
#include <iostream>

using namespace std;

class Shape {

public:

   virtual void getData() {}

   virtual void area() {

      cout << "Area of shape.\n";

   }

};

class Circle : public Shape {

   float radius;

public:

   void getData() override {

      cout << "Enter radius of circle: ";

      cin >> radius;

   }


   void area() override {

      float a = 3.14 * radius * radius;

      cout << "Area of Circle: " << a << endl;

   }
```

```
};
class Rectangle : public Shape {
    float length, breadth;
public:
    void getData() override {
        cout << "Enter length and breadth of rectangle: ";
        cin >> length >> breadth;
    }
    void area() override {
        float a = length * breadth;
        cout << "Area of Rectangle: " << a << endl;
    }
};
int main() {
    Circle c;
    Rectangle r;
    Shape* s;
    s = &c;
    s->getData();
    s->area();
    s = &r;
    s->getData();
    s->area();
    return 0;
}
```

Output:

```
Enter radius of circle: 5
Area of Circle: 78.5
Enter length and breadth of rectangle: 4 6
Area of Rectangle: 24
```

Q45. Create a base class Student that stores roll number.Derive two classes:Test (marks in 5 subjects)

Sports (sports score).Create another derived class Result from both Test and Sports.

Add a method to calculate total score (sum of subject marks and sports score)

Display roll number, all marks, sports score, and total.

Objective:

Use multiple inheritance to combine academic test scores and sports score to calculate and display total result.

Description:

- Student class stores roll number.
- Test (inherits Student) stores marks of 5 subjects.
- Sports stores sports score.
- Result inherits from both Test and Sports, calculates total, and displays all data.

Code:

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

class Student {
protected:
   int rollNo;

public:
   void inputRoll() {
      cout << "Enter Roll Number: ";
      cin >> rollNo;
   }
   void displayRoll() {
      cout << "Roll Number: " << rollNo << endl;
   }
};
class Test : public Student {
protected:
   int marks[5];
```

```
public:
    void inputMarks() {
        cout << "Enter marks for 5 subjects: ";
        for (int i = 0; i < 5; i++) {
            cin >> marks[i];
        }
    }
    void displayMarks() {
        cout << "Marks: ";
        for (int i = 0; i < 5; i++) {
            cout << marks[i] << " ";
        }
        cout << endl;
    }
};
class Sports {
protected:
    int sportsScore;
public:
    void inputSports() {
        cout << "Enter Sports Score: ";
        cin >> sportsScore;
    }
    void displaySports() {
        cout << "Sports Score: " << sportsScore << endl;
    }
};

class Result : public Test, public Sports {
    int total;
public:
    void calculateTotal() {
```

```
        total = sportsScore;
        for (int i = 0; i < 5; i++) {
            total += marks[i];
        }
    }
    void displayResult() {
        displayRoll();
        displayMarks();
        displaySports();
        cout << "Total Score (Marks + Sports): " << total << endl;
    }
};
int main() {
    Result r;
    r.inputRoll();
    r.inputMarks();
    r.inputSports();
    r.calculateTotal();
    cout << "\n--- Student Result ---\n";
    r.displayResult();

    return 0;
}
```

Output:

```
Enter Roll Number: 101
Enter marks for 5 subjects: 80 75 90 85 70
Enter Sports Score: 40
--- Student Result ---
Roll Number: 101
Marks: 80 75 90 85 70
Sports Score: 40
Total Score (Marks + Sports): 440
```

Q46.WAP to count the number of objects created.

Objective:

Write a program to count how many objects of a class are created using a static variable.

Description:

This program uses a **static data member** in the class to count the number of times the constructor is called (i.e., how many objects are created). Static variables are shared across all objects of the class.

Code:

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

class CountObjects {
    static int count;
public:
    CountObjects() {
        count++;
    }
    static void showCount() {
        cout << "Total objects created: " << count << endl;
    }
};
int CountObjects::count = 0;
int main() {
    CountObjects obj1, obj2, obj3;
    CountObjects::showCount();

    CountObjects obj4;
    CountObjects::showCount();
    return 0;
}
```

Output:

```
Total objects created: 3
Total objects created: 4
```

Q47.Create two objects for a class timer. Assume three data members' hours, minutes, and seconds. Use two constructors for initializing the objects. Add two-time objects using operator overloading. Display appropriate values of hours, minutes, and seconds after addition.

Objective

Use constructor overloading and operator overloading to add two time objects (hours, minutes, seconds) and display the result.

Description

- Class Timer has data members: hours, minutes, seconds.
- Uses two constructors to initialize: one default and one parameterized.
- Overloads the + operator to add two Timer objects.
- Handles carry-over for seconds minutes and minutes hours.

Code:

```
#include <iostream>
using namespace std;
class Timer {
   int hours, minutes, seconds;

public:
   Timer() {
      hours = minutes = seconds = 0;
   }
   Timer(int h, int m, int s) {
      hours = h;
      minutes = m;
      seconds = s;
   }
   Timer operator+(Timer t) {
      Timer temp;
      temp.seconds = seconds + t.seconds;
      temp.minutes = minutes + t.minutes + (temp.seconds / 60);
      temp.seconds %= 60;
      temp.hours = hours + t.hours + (temp.minutes / 60);
      temp.minutes %= 60;
```

```
        return temp;
    }
    void display() {
        cout << "Time: " << hours << "h " << minutes << "m " << seconds << "s\n";
    }
};
int main() {
    Timer t1(2, 45, 50);
    Timer t2(1, 30, 20);

    Timer t3;
    t3 = t1 + t2;

    cout << "First Time: "; t1.display();
    cout << "Second Time: "; t2.display();
    cout << "Total Time: "; t3.display();

    return 0;
}
```

Output:

First Time: Time: 2h 45m 50s

Second Time: Time: 1h 30m 20s

Total Time: Time: 4h 16m 10s

Q48.WAP to demonstrate constructor overloading.

Objective:

To demonstrate constructor overloading by creating multiple constructors with different parameters in a class.

Description:

Constructor overloading allows a class to have more than one constructor with different signatures. In this program, a class Demo has:

- a default constructor
- a parameterized constructor
- a two-parameter constructor

Code:

```
#include <iostream>

using namespace std;


class Demo {
   int a, b;


public:


   Demo() {
      a = 0;
      b = 0;
      cout << "Default Constructor: a = " << a << ", b = " << b << endl;
   }


   Demo(int x) {
      a = x;
      b = 0;
      cout << "One-parameter Constructor: a = " << a << ", b = " << b << endl;
   }


   Demo(int x, int y) {
      a = x;
```

```
        b = y;
        cout << "Two-parameter Constructor: a = " << a << ", b = " << b << endl;
    }
};
int main() {
    Demo d1;
    Demo d2(5);
    Demo d3(3, 7);

    return 0;
}
```

Output:

Default Constructor: a = 0, b = 0

One-parameter Constructor: a = 5, b = 0

Two-parameter Constructor: a = 3, b = 7

Q49.WAP to copy the content of one file to another.

Objective:

To copy the contents of one file into another using file handling in C++.

Description:

This program uses file streams (ifstream and ofstream) to:

- Open a source file in read mode.
- Open a destination file in write mode.
- Read characters from the source and write them to the destination.

Code:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
   char ch;
   ifstream source("input.txt");
   ofstream dest("output.txt");

   if (!source) {
      cout << "Source file could not be opened.\n";
      return 1;
   }
   while (source.get(ch)) {
      dest.put(ch);
   }
   cout << "File copied successfully.\n";

   source.close();
   dest.close();
   return 0;
}
```

Output:

File copied successfully.

Q50.WAP to update the content of a file.

Objective:

To update the contents of an existing file using file handling in C++.

Description:

This program opens an existing file and appends new content to it using ofstream in append mode (ios::app). This is one way to update the file without erasing existing data.

Code:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    string text;
    ofstream file;

    file.open("update.txt", ios::app);

    if (!file) {
        cout << "Error opening file.\n";
        return 1;
    }
    cout << "Enter text to add to the file: ";
    getline(cin, text);

    file << text << endl;

    cout << "File updated successfully.\n";
    file.close();
    return 0;
}
```

Output

Enter text to add to the file: This is the new line added.

File updated successfully.