1. Write a C++ Program to print the following using cout and manupulators (endl, left, right, setw)

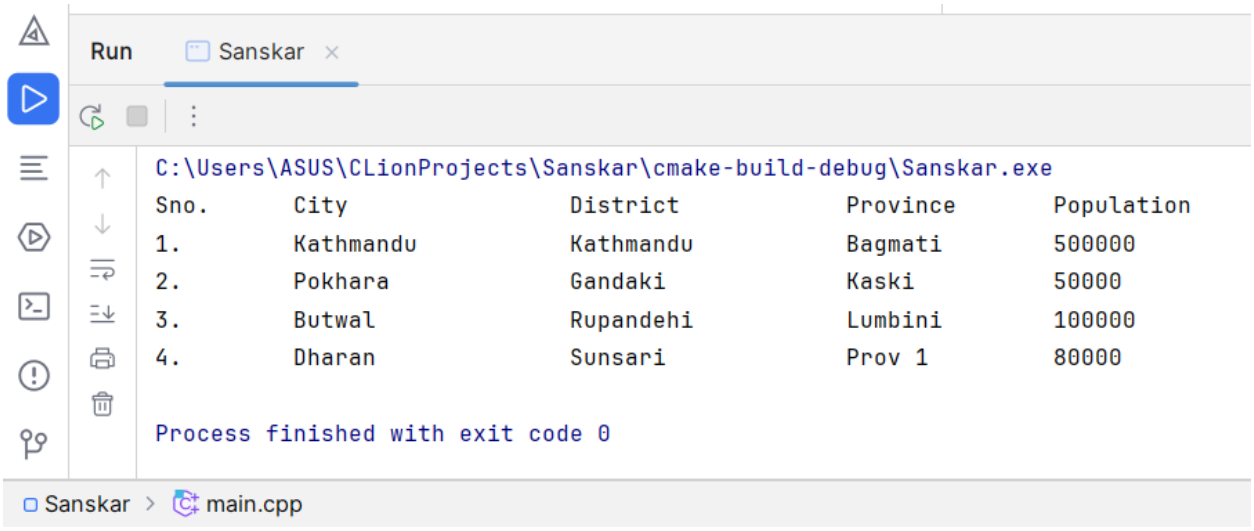
S.No.	City	District	Province	Population
1.	KATHMANDU	Kathmandu	Bagmati	500000
2.	Pokhara	Kaski	Gandaki	50000
3.	Butwal	Rupendehi	Lumbini	100000
4.	Dharan	Sunsari	Prov 1	80000

Program:

```
#include <iostream>

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

int main() {
  cout << left << setw(10) << "Sno."
       << setw(20) << "City"
       << setw(20) << "District"
       << setw(15) << "Province"
       << setw(12) << "Population" << endl;

  cout << left << setw(10) << "1."
       << setw(20) << "Kathmandu"
       << setw(20) << "Kathmandu"
       << setw(15) << "Bagmati"
       << setw(12) << "500000" << endl;

  cout << left << setw(10) << "2."
       << setw(20) << "Pokhara"
       << setw(20) << "Gandaki"
       << setw(15) << "Kaski"
       << setw(12) << "50000" << endl;


  cout << left << setw(10) << "3."
       << setw(20) << "Butwal"
       << setw(20) << "Rupandehi"
       << setw(15) << "Lumbini"
       << setw(12) << "100000" << endl;
```

```
    cout << left << setw(10) << "4."
         << setw(20) << "Dharan"
         << setw(20) << "Sunsari"
         << setw(15) << "Prov 1"
         << setw(12) << "80000" << endl;

    return 0;
}
```

Output:

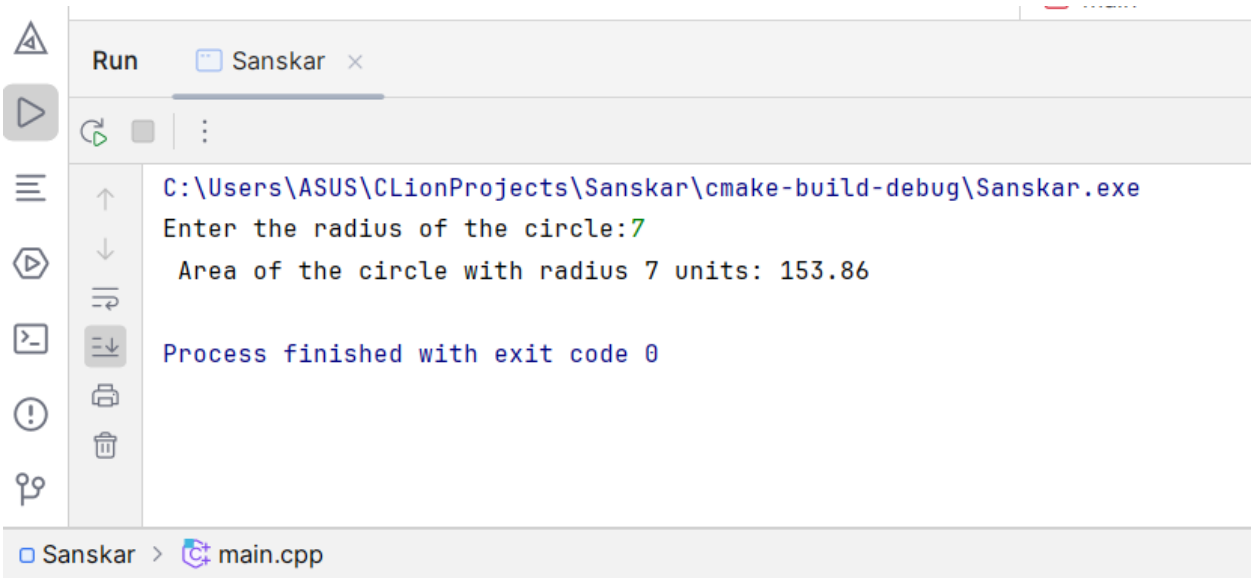


2. Write a Program defining an inline function to compute the area of circle with radius as input.

Program:

```
#include <iostream>
#define PI 3.14
using namespace std;
// Define an inline function to compute the area of a circle
inline double computeCircleArea(double radius) {
  return PI * radius * radius;
}

int main() {
  double radius;
  cout << "Enter the radius of the circle: ";
  cin >> radius;

  double area = computeCircleArea(radius);
  cout << "Area of the circle with radius " << radius << " units: " << area << std::endl;

  return 0;
}
```

Output:



3. Write a program to input a string and print the string and its reverse. Define your own function to reverse the string in your program.

Program:

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

void reverseString(string& str) {
   int n = str.length();
   for (int i = 0; i < n / 2; ++i) {
      // Swap characters from both ends
      char temp = str[i];
      str[i] = str[n - i - 1];
      str[n - i - 1] = temp;
   }
}
int main() {
   string inputString;
   cout << "Enter a string: ";
   cin >> inputString;

   cout << "Original string: " << inputString << endl;
      reverseString(inputString);

   cout << "Reversed string: " << inputString << endl;

   return 0;
}
```

Output:

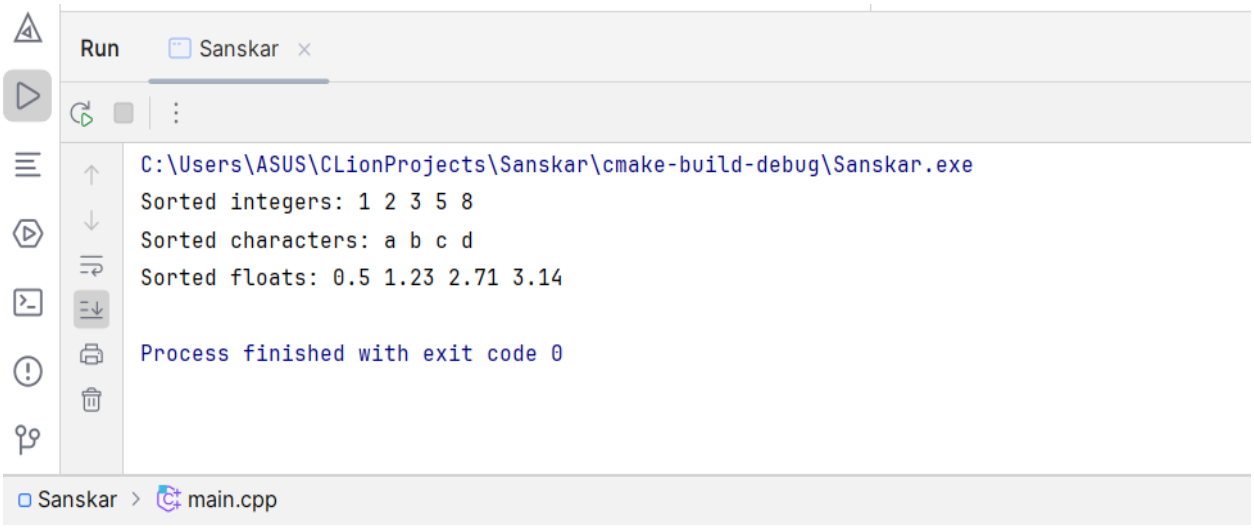
4. Write a program overloading function sort() to sort an array of integers, characters as well as floating point numbers(Use any sorting algorithm).

Program:

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

// Function to sort an array of integers using Bubble Sort
void bubbleSort(int arr[], int size) {
  for (int i = 0; i < size - 1; ++i) {
    for (int j = 0; j < size - i - 1; ++j) {
      if (arr[j] > arr[j + 1]) {
        // Swap elements
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}

// Function to sort an array of characters using Bubble Sort
void bubbleSort(char arr[], int size) {
  for (int i = 0; i < size - 1; ++i) {
    for (int j = 0; j < size - i - 1; ++j) {
      if (arr[j] > arr[j + 1]) {
        // Swap characters
        char temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}
```

```
// Function to sort an array of floating-point numbers using Bubble Sort
void bubbleSort(float arr[], int size) {
  for (int i = 0; i < size - 1; ++i) {
    for (int j = 0; j < size - i - 1; ++j) {
      if (arr[j] > arr[j + 1]) {
        // Swap floats
        float temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}



int main() {
  // Example: Sorting an array of integers
  int intArray[] = {5, 2, 8, 1, 3};
  int intSize = sizeof(intArray) / sizeof(int);
  bubbleSort(intArray, intSize);
  cout << "Sorted integers: ";
  for (int i = 0; i < intSize; ++i)
    cout << intArray[i] << " ";
  cout << endl;

  char charArray[] = {'d', 'a', 'c', 'b'};
  int charSize = sizeof(charArray) / sizeof(char);
  bubbleSort(charArray, charSize);
  cout << "Sorted characters: ";
  for (int i = 0; i < charSize; ++i)
    cout << charArray[i] << " ";
  cout << endl;

  float floatArray[] = {3.14, 1.23, 2.71, 0.5};
  int floatSize = sizeof(floatArray) / sizeof(float);
  bubbleSort(floatArray, floatSize);
  cout << "Sorted floats: ";
  for (int i = 0; i < floatSize; ++i)
    cout << floatArray[i] << " ";
  cout << endl;

  return 0;
}
```

Output:


Run
Sanskar
C:\Users\ASUS\CLionProjects\Sanskar\cmake-build-debug\Sanskar.exe
Sorted integers: 1 2 3 5 8
Sorted characters: a b c d
Sorted floats: 0.5 1.23 2.71 3.14
Process finished with exit code 0
Sanskar
main.cpp

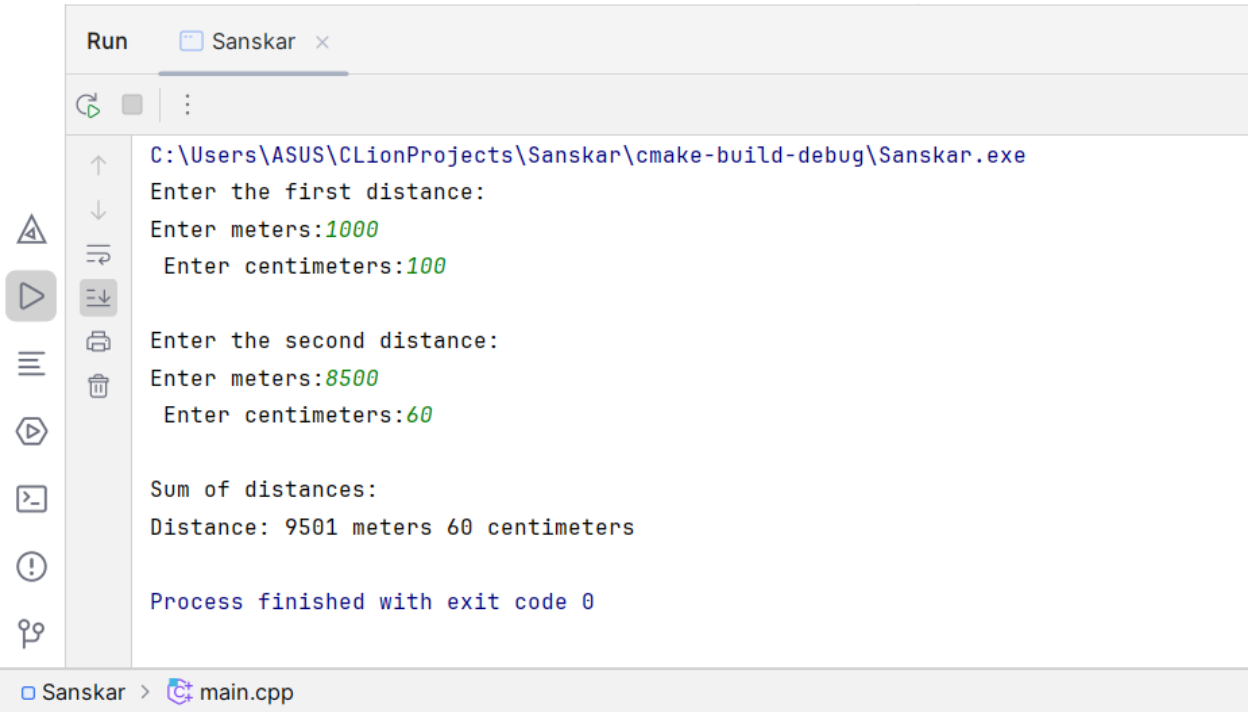
5. Write a program that uses a structure Distance with data members meter and centimeter. Add functions in structure to take input and output as well as the function to add the two variables of Distance and return the sum. Your program should display the result.

Program:

```
#include <iostream>

using namespace std;


// Structure to represent distance in meters and centimeters

struct Distance {

  int meters;

  float centimeters;

};


// Function to input distance

void inputDistance(Distance& d) {

  cout << "Enter meters: ";

  cin >> d.meters;

  cout << "Enter centimeters: ";

  cin >> d.centimeters;

}


// Function to display distance

void displayDistance(const Distance& d) {

  cout << "Distance: " << d.meters << " meters " << d.centimeters << " centimeters" << endl;

}


// Function to add two distances

Distance addDistances(const Distance& d1, const Distance& d2) {

  Distance sum;
```

```
    sum.meters = d1.meters + d2.meters;
    sum.centimeters = d1.centimeters + d2.centimeters;

    // Adjust centimeters if greater than 100
    if (sum.centimeters >= 100) {
        sum.meters += static_cast<int>(sum.centimeters / 100);
        sum.centimeters -= static_cast<int>(sum.centimeters / 100) * 100;
    }

    return sum;
}

int main() {
    Distance distance1, distance2;

    cout << "Enter the first distance:" << endl;
    inputDistance(distance1);

    cout << "\nEnter the second distance:" << endl;
    inputDistance(distance2);

    Distance totalDistance = addDistances(distance1, distance2);

    cout << "\nSum of distances:" << endl;
    displayDistance(totalDistance);

    return 0;
}
```

Output:


Run
Sanskar
C:\Users\ASUS\CLionProjects\Sanskar\cmake-build-debug\Sanskar.exe
Enter the first distance:
Enter meters:1000
Enter centimeters:100
Enter the second distance:
Enter meters:8500
Enter centimeters:60
Sum of distances:
Distance: 9501 meters 60 centimeters
Process finished with exit code 0
Sanskar
main.cpp

6. Write program with objects as function(to add time) argument by passing by value, passing by address and passing by reference defining a class Time with data member hour, minute, second as integers. Write member functions to read the data for objects and to show the value of objects of Time.

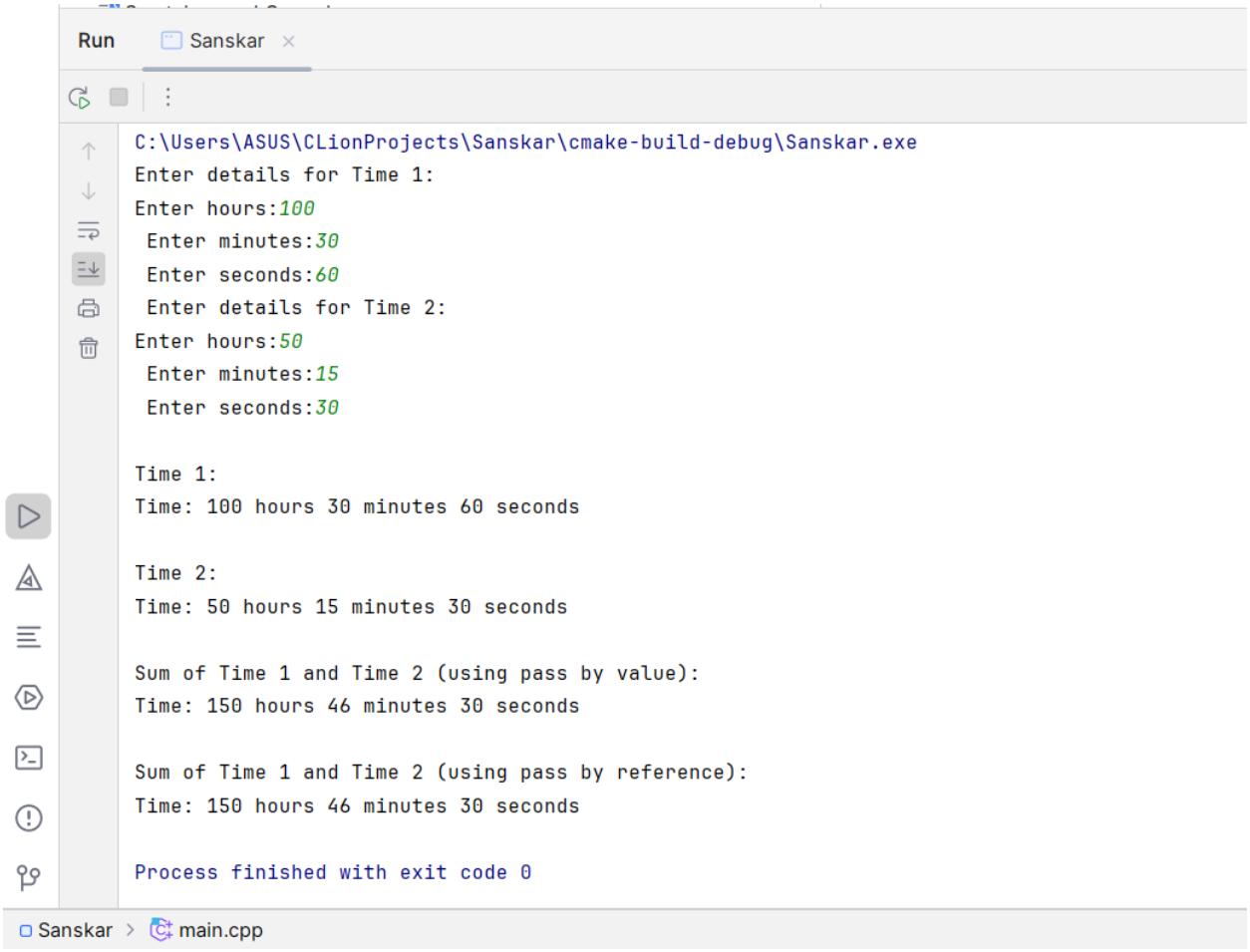
Program:

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

class Time {
private:
  int hour;
  int minute;
  int second;

public:

  void readTime() {
    cout << "Enter hours: ";
    cin >> hour;
    cout << "Enter minutes: ";
    cin >> minute;
    cout << "Enter seconds: ";
    cin >> second;
  }

  void displayTime() {
    cout << "Time: " << hour << " hours " << minute << " minutes " << second <<" seconds" << endl;
  }

  Time addByValue(Time t) {
    Time sum;
    sum.hour = hour + t.hour;
    sum.minute = minute + t.minute;
    sum.second = second + t.second;

    if (sum.second >= 60) {
      sum.minute += sum.second / 60;
      sum.second = sum.second % 60;
    }

    if (sum.minute >= 60) {
      sum.hour += sum.minute / 60;
      sum.minute = sum.minute % 60;
```

```
    }

    return sum;
  }

  void addByReference(Time &t) {
    hour += t.hour;
    minute += t.minute;
    second += t.second;

    if (second >= 60) {
      minute += second / 60;
      second = second % 60;
    }

    if (minute >= 60) {
      hour += minute / 60;
      minute = minute % 60;
    }
  }
};

int main() {
  Time time1, time2;

  cout << "Enter details for Time 1:" << endl;
  time1.readTime();

  cout << "Enter details for Time 2:" << endl;
  time2.readTime();

  cout << "\nTime 1:" << endl;
  time1.displayTime();
  cout << "\nTime 2:" << endl;
  time2.displayTime();

  Time sumByValue = time1.addByValue(time2);
  cout << "\nSum of Time 1 and Time 2 (using pass by value):" << endl;
  sumByValue.displayTime();
```

```
    time1.addByReference(time2);
    cout << "\nSum of Time 1 and Time 2 (using pass by reference):" << endl;
    time1.displayTime();

    return 0;
}
```

Output:



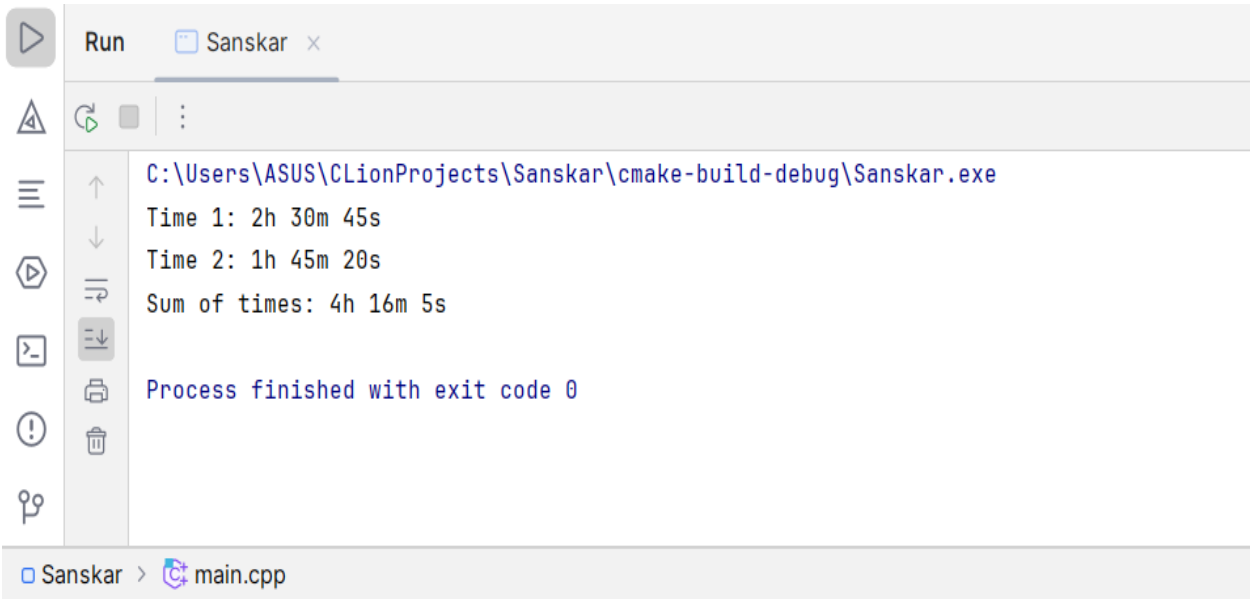
7. Define a friend function addtime() with objects as arguments and return the sum of two objects. Show the values of each object and their sum as output.

Program:

```
#include <iostream>

using namespace std;


class Time {

private:

  int hours;

  int minutes;

  int seconds;


public:

  Time(int h = 0, int m = 0, int s = 0) : hours(h), minutes(m), seconds(s) {}


  // Declare the friend function

  friend Time addTime(const Time& t1, const Time& t2);


  void display() const {

    cout << hours << "h " << minutes << "m " << seconds << "s";

  }

};


// Define the friend function

Time addTime(const Time& t1, const Time& t2) {

  Time sum;

  sum.hours = t1.hours + t2.hours;

  sum.minutes = t1.minutes + t2.minutes;

  sum.seconds = t1.seconds + t2.seconds;
```

```
    // Adjust minutes and seconds if they exceed 60
    sum.minutes += sum.seconds / 60;
    sum.seconds %= 60;
    sum.hours += sum.minutes / 60;
    sum.minutes %= 60;

    return sum;
}

int main() {
    Time time1(2, 30, 45);
    Time time2(1, 45, 20);

    cout << "Time 1: ";
    time1.display();
    cout << endl;

    cout << "Time 2: ";
    time2.display();
    cout << endl;

    Time sum = addTime(time1, time2);
    cout << "Sum of times: ";
    sum.display();
    cout << endl;

    return 0;
}
```

Output:


Run
Sanskar
C:\Users\ASUS\CLionProjects\Sanskar\cmake-build-debug\Sanskar.exe
Time 1: 2h 30m 45s
Time 2: 1h 45m 20s
Sum of times: 4h 16m 5s
Process finished with exit code 0
Sanskar
main.cpp

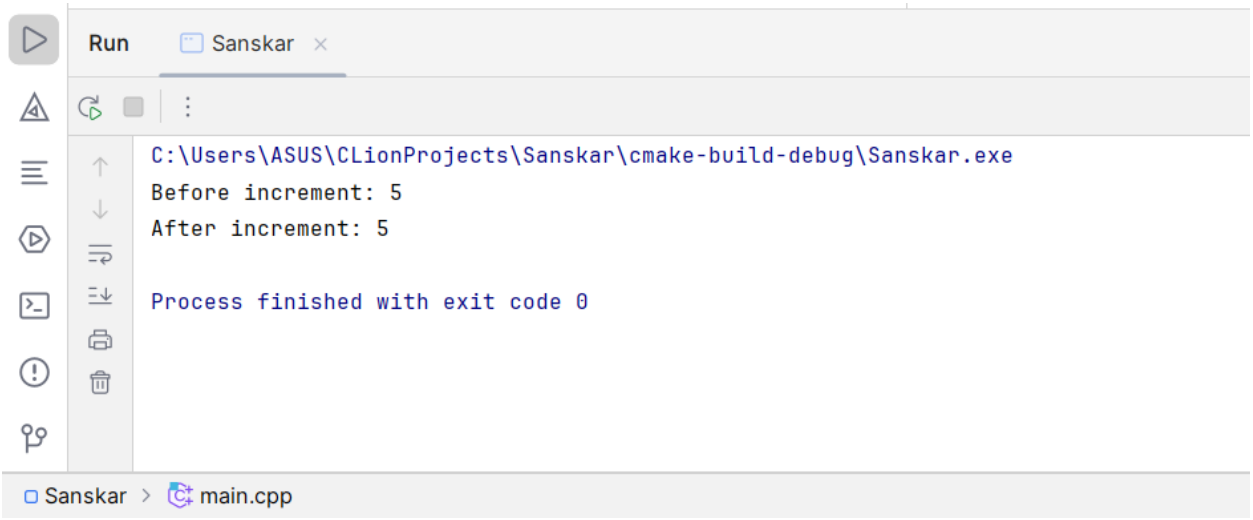
8. Write different programs to implement passing by reference and passing by value in C++.

Pass by Value

Program:

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

// Function to increment a value by 1 (pass by value)
void incrementByValue(int num) {
  num++;
}

int main() {
  int x = 5;
  cout << "Before increment: " << x << endl;
  incrementByValue(x);
  cout << "After increment: " << x << endl;
  return 0;
}
```

Output:

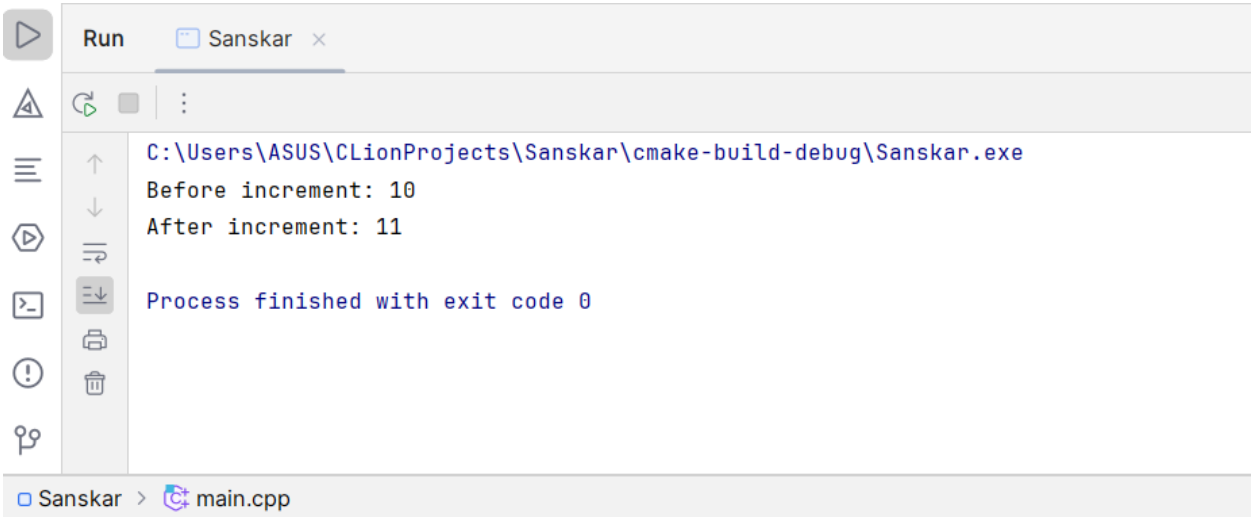


Pass by Reference

Program:

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

// Function to increment a value by 1 (pass by reference)
void incrementByReference(int& num) {
    num++;
}

int main() {
    int y = 10;
    cout << "Before increment: " << y << endl;
    incrementByReference(y);
    cout << "After increment: " << y << endl;
    return 0;
}
```

Output:



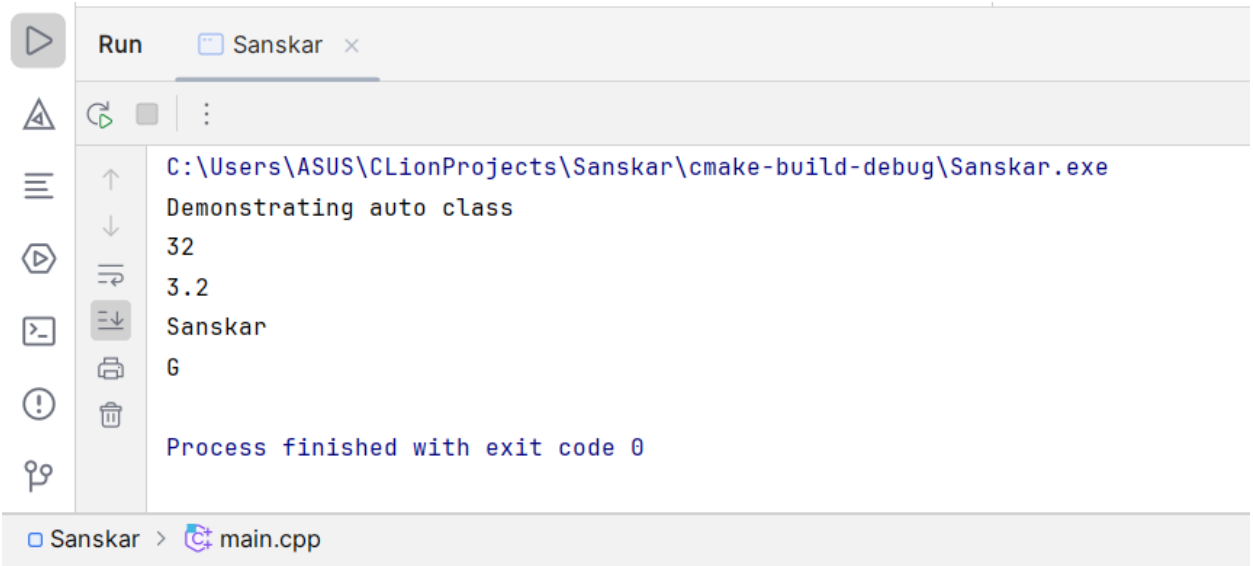
9. Write different programs to implement different storage classes (auto, register, extern and static) in C++ with its output.

Auto

Program:

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

void autoStorageClass() {
   cout << "Demonstrating auto class\n";
   int a = 32;
   float b = 3.2;
   char* c = "Sanskar";
   char d = 'G';

   cout << a << " \n";
   cout << b << " \n";
   cout << c << " \n";
   cout << d << " \n";
}

int main() {
   autoStorageClass();
   return 0;
}
```

Output:

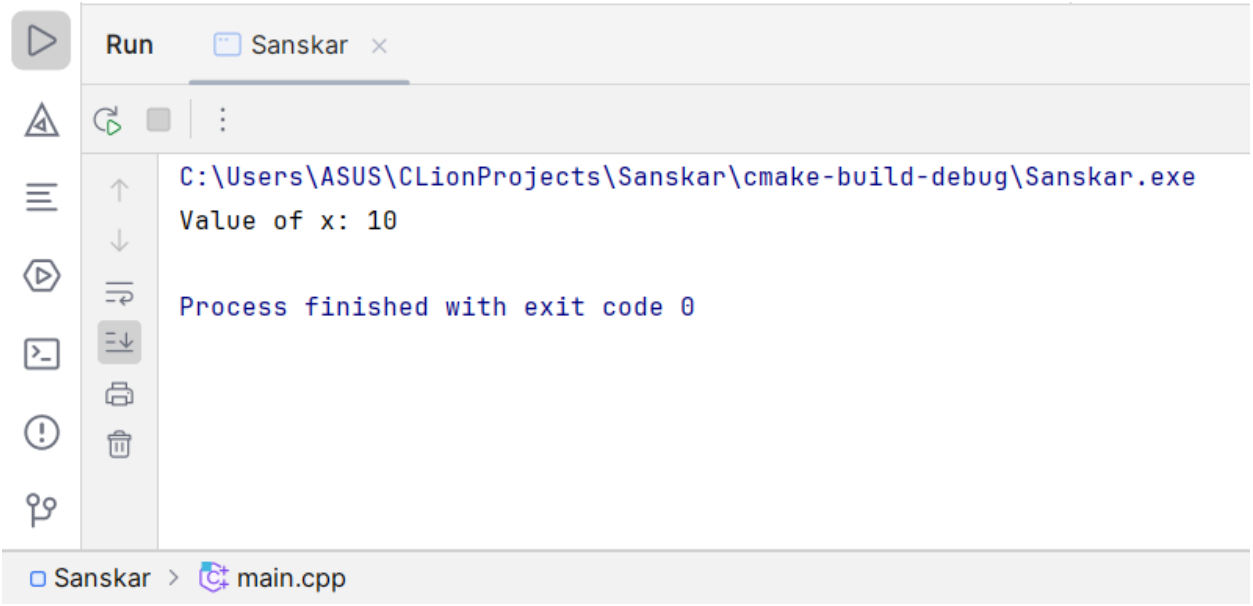


Register

Program:

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

int main() {
   register int x = 10;
   cout << "Value of x: " << x << endl;
   return 0;
}
```

Output:

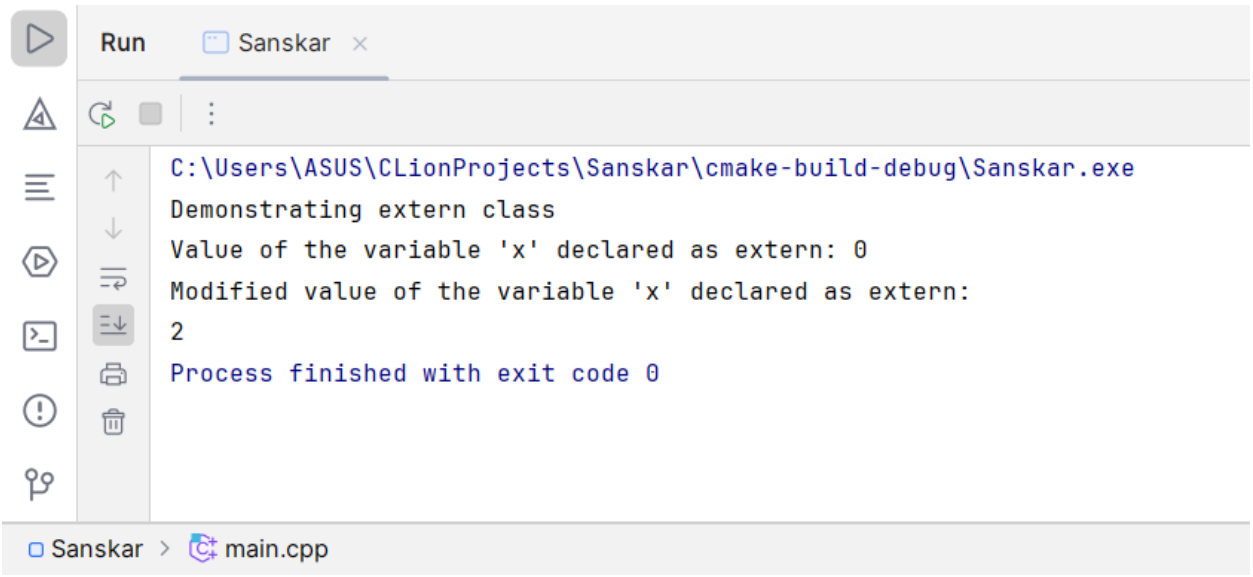


Extern

Program:

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

int x; // Declare an extern variable
void externStorageClass() {
   cout << "Demonstrating extern class\n";
   extern int x; // Use the extern variable
   cout << "Value of the variable 'x' declared as extern: " << x << "\n";
   x = 2; // Modify the value of the extern variable
   cout << "Modified value of the variable 'x' declared as extern: \n" << x;
}

int main() {
   externStorageClass();
   return 0;
}
```

Output:

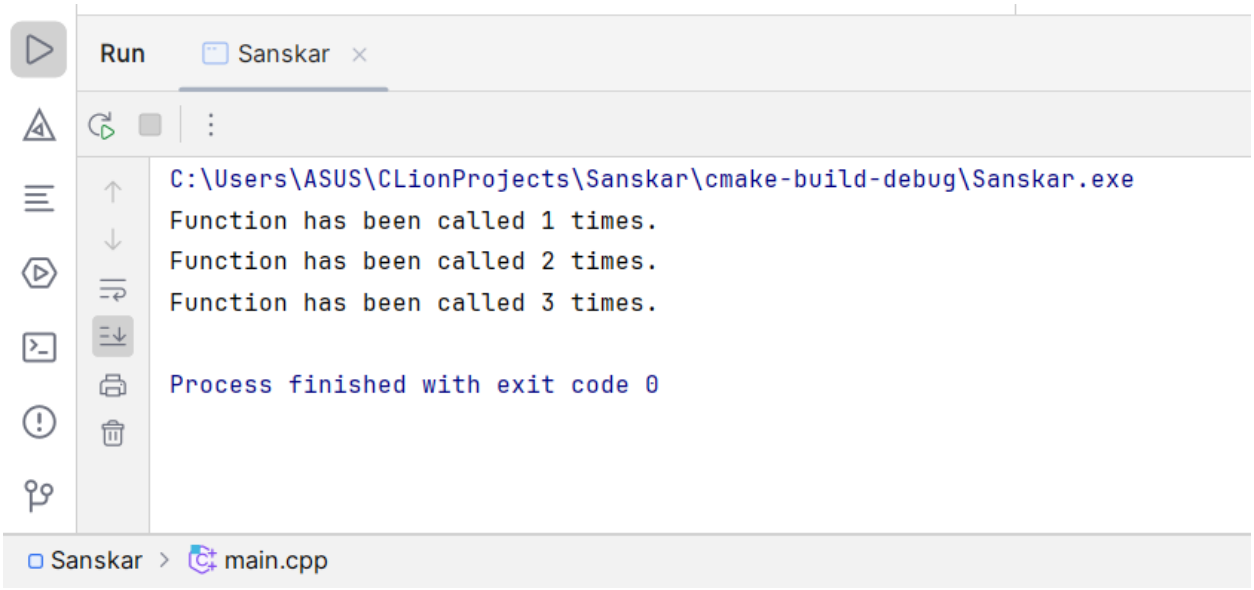


Static

Program:

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

void countCalls() {
   static int counter = 0; // Static variable
   counter++;
   cout << "Function has been called " << counter << " times.\n";
}

int main() {
   countCalls();
   countCalls();
   countCalls();
   return 0;
}
```

Output:



10. Write a C++ program to illustrate dynamic allocation and de-allocation of memory using new and delete.

Program:

```
#include <iostream>

using namespace std;


int main() {
   // Allocate memory for an integer dynamically
   int* ptr = new int;
   *ptr = 10; // Assign a value to the dynamically allocated memory


   cout << "Address: " << ptr << endl;
   cout << "Value: " << *ptr << endl;


   // Deallocate the memory
   delete ptr;
   return 0;
}
```

Output:

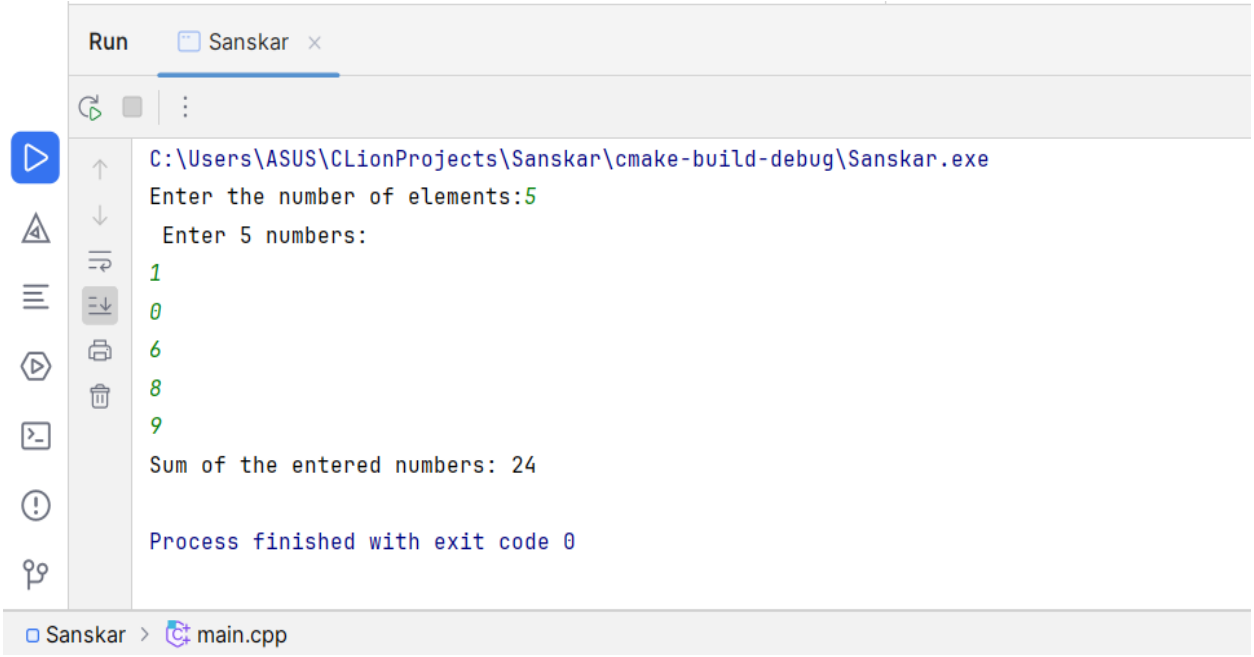
11. Write a program using dynamic memory allocation to get input an array of numbers and find the sum of N numbers stored in the array using a function to compute the sum.

Program:

```
#include <iostream>

using namespace std;


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

   // Allocate memory for an array of integers dynamically
   int* arr = new int[n];

   // Input elements
   cout << "Enter " << n << " numbers:\n";
   for (int i = 0; i < n; ++i) {
      cin >> arr[i];
   }

   // Calculate the sum
   int sum = 0;
   for (int i = 0; i < n; ++i) {
      sum += arr[i];
   }

   cout << "Sum of the entered numbers: " << sum << endl;
```

```
    // Deallocate the memory
    delete[] arr;

    return 0;
}
```

Output:

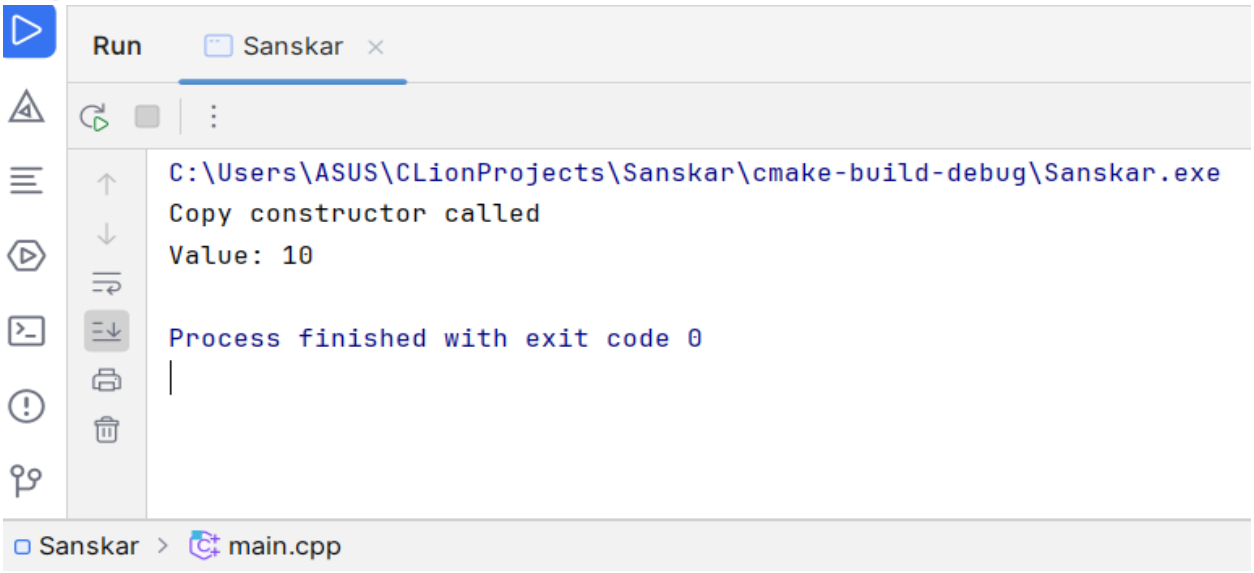


12. Write a program to implement user defined constructor and copy constructor.

Program:

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

class MyClass {
private:
  int value;
public:
  // User-defined constructor
  MyClass(int v) : value(v) {}
  // Copy constructor
  MyClass(const MyClass& other) {
    value = other.value;
    cout << "Copy constructor called" << endl;
  }

  void display() {
    cout << "Value: " << value << endl;
  }
};
int main() {
  MyClass original(10);
  MyClass copy(original); // Create a copy using the copy constructor
  copy.display();
  return 0;
}
```

Output:



13. Write a program to illustrate constructor overloading in C++.

Program:

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

class Shape {
public:
  virtual double calculateArea() const = 0;
};

class Rectangle : public Shape {
private:
  double length;
  double width;

public:
  Rectangle(double l, double w) : length(l), width(w) {}

  double calculateArea() const override {
    return length * width;
  }
};

class Square : public Shape {
private:
  double side;

public:
  Square(double s) : side(s) {}

  double calculateArea() const override {
    return side * side;
  }
};

class Circle : public Shape {
private:
  double radius;

public:
  Circle(double r) : radius(r) {}
```

```
    double calculateArea() const override {
        return 3.14159 * radius * radius;
    }
};

int main() {
    Rectangle rect(5.0, 3.0);
    Square square(4.0);
    Circle circle(2.5);

    cout << "Area of Rectangle: " << rect.calculateArea() << endl;
    cout << "Area of Square: " << square.calculateArea() << endl;
    cout << "Area of Circle: " << circle.calculateArea() << endl;

    return 0;
}
```

Output: