Q5. Write a C++ program to display "Hello World" on the screen.
The program should print the message "Hello World" as output.

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

int main() {
    cout << "Hello World\n";
    return 0;
}
```

SAMPLE OUTPUT:

Hello World

Q6. Write a C++ program to take a line of text as input using getline() and display it back to the user.
The program should read a full line of input (including spaces) and then print it.

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

int main() {
    string text;
    cout << "Enter a line of text:\n";
    getline(cin, text);

    cout << "You entered: " << text << "\n";
    return 0;
}
```

SAMPLE OUTPUT:

Enter a line of text:
I am learning C++
You entered: I am learning C++

Q7. You are given an array of elements. Now you need to choose the best index of this array. An index of the array is called best if the special sum of this index is maximum across the special sums of all the other indices.

To calculate the special sum for any index:
> • Pick the first element at that index and add it to your sum.
> • Then pick the next two elements and add them to your sum.
> • Then pick the next three elements and add them to your sum.
> • Continue this process, increasing the group size by 1 each time, until it is no longer possible to pick the required number of elements.

Find the best index and, in the output, print its corresponding special sum.

Note:
> • There may be more than one best index, but you only need to print the maximum special sum.

Input:
> • First line contains an integer — the number of elements in the array.
> • Next line contains space-separated integers denoting the elements of the array.

Output:
> •Print an integer that denotes the maximum special sum.

Input/Output Format	
Typical Input	**Expected Output**
5 1 3 1 2 5	8
10 2 1 3 9 2 4 -10 -9 1 3	9

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

int main() {
    int n;
    cin >> n;

    int arr[1000];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    int maxSum = -1000000;

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

```cpp
        int groupSize = 1;
        int index = i;

        while (index + groupSize - 1 < n) {
            for (int j = 0; j < groupSize; j++) {
                sum += arr[index + j];
            }
            index += groupSize;
            groupSize++;
        }

        if (sum > maxSum) {
            maxSum = sum;
        }
    }

    cout << maxSum << endl;
    return 0;
}
```

SAMPLE OUTPUT:

5
1 2 3 4 5

11

Q8. A cinema hall wants to develop a small program in C++ seating arrangement for cinema. The hall contains 5 rows (10 seats each) = 50 seats available. The cinema hall management must have full functionality:
1. **Initialization of seats**
2. **Booking a seat**
3. **Cancel booking**
4. **Display the seating arrangement**

You need to design a structure for cinema hall. It contains (2D array) for seating arrangement. Additional data to be maintained:
- **Hall name**
- **Total seats**
- **Booking seat count**

Note:
- **Pass the structures by reference to a function for operations like initialization, display, cancel, booking.**
- **Write a perfect C++ program using appropriate arrays and structures.**
- **Must be practical for a mini project.**

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

struct CinemaHall {
    string hallName;
    int totalSeats;
    int bookedSeats;
    int seats[5][10];
};

void initializeSeats(CinemaHall &hall) {
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 10; j++) {
            hall.seats[i][j] = 0;
        }
    }
    hall.bookedSeats = 0;
}

void bookSeat(CinemaHall &hall, int row, int col) {
    if (row < 0 || row >= 5 || col < 0 || col >= 10) {
        cout << "Invalid seat position!\n";
        return;
    }
    if (hall.seats[row][col] == 0) {
        hall.seats[row][col] = 1;
        hall.bookedSeats++;
        cout << "Seat booked successfully!\n";
    } else {
        cout << "Seat already booked!\n";
    }
```

```cpp
}

void cancelSeat(CinemaHall &hall, int row, int col) {
    if (row < 0 || row >= 5 || col < 0 || col >= 10) {
        cout << "Invalid seat position!\n";
        return;
    }
    if (hall.seats[row][col] == 1) {
        hall.seats[row][col] = 0;
        hall.bookedSeats--;
        cout << "Booking cancelled!\n";
    } else {
        cout << "Seat was not booked!\n";
    }
}

void displaySeats(const CinemaHall &hall) {
    cout << "\nSeating Arrangement \n";
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 10; j++) {
            cout << hall.seats[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Total Seats: " << hall.totalSeats
         << " | Booked: " << hall.bookedSeats
         << " | Available: " << hall.totalSeats - hall.bookedSeats << endl;
}

int main() {
    CinemaHall hall;
    hall.hallName = "Global Cinema";
    hall.totalSeats = 50;
    hall.bookedSeats = 0;

    int choice, row, col;

    do {
        cout << "\n--- Cinema Hall Menu ---\n";
        cout << "1. Initialize Seats\n";
        cout << "2. Book a Seat\n";
        cout << "3. Cancel Booking\n";
        cout << "4. Display Seats\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                initializeSeats(hall);
```

```cpp
                break;
            case 2:
                cout << "Enter row (0-4) and column (0-9): ";
                cin >> row >> col;
                bookSeat(hall, row, col);
                break;
            case 3:
                cout << "Enter row (0-4) and column (0-9): ";
                cin >> row >> col;
                cancelSeat(hall, row, col);
                break;
            case 4:
                displaySeats(hall);
                break;
            case 5:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Invalid choice!\n";
        }
    } while (choice != 5);

    return 0;
}
```

SAMPLE OUTPUT:
--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 1

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 2
Enter row (0-4) and column (0-9): 2 3
Seat booked successfully!

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat

3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 2
Enter row (0-4) and column (0-9): 2 3
Seat already booked!

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 4

Seating Arrangement
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Total Seats: 50 | Booked: 1 | Available: 49

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 3
Enter row (0-4) and column (0-9): 2 3
Booking cancelled!

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 5
Exiting...

Q9. Design a simple banking system using classes and encapsulation in C++.
Create a class bankAcccount with the following private data members:
 1. **accountNumber**
 2. **accountHolderName**
 3. **balance**
Implement public methods to provide controlled access.
 1. **Set AC Details: In Ac Details, set account number, name and balance**
 2. **Deposit:**
 3. **Withdraw**
 4. **getBalance**
 5. **Display Account Info**
Ensure data hiding, by keeping sensitive details like balance, name private and only accessible through methods.
Create multiple accounts using an array of objects, allow user to perform operations like deposit, withdraw, and show balance on these accounts.
Do not use constructors, instead initialize data through setter functions.

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

class bankAccount {
private:
    int accountNumber;
    string accountHolderName;
    double balance;

public:
    void setAccountDetails(int no, string name, double bal) {
        accountNumber = no;
        accountHolderName = name;
        balance = bal;
    }

    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            cout << "Amount deposited successfully.\n";
        } else {
            cout << "Enter a valid amount.\n";
        }
    }

    void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            cout << "Amount withdrawn successfully.\n";
        } else {
            cout << "Insufficient balance or invalid amount.\n";
        }
```

```cpp
    }

    void displayAccountInfo(int no) {
        if (no == accountNumber) {
            cout << "---------------------------------\n";
            cout << "Account Number: " << accountNumber << endl;
            cout << "Account Holder Name: " << accountHolderName << endl;
            cout << "Available Balance: " << balance << endl;
            cout << "---------------------------------\n";
        } else {
            cout << "Invalid Account Number.\n";
        }
    }

    double getBalance(int no) {
        if (no == accountNumber) {
            return balance;
        } else {
            cout << "Invalid Account Number.\n";
            return -1;
        }
    }

    int getAccountNumber() {
        return accountNumber;
    }
};

int main() {
    bankAccount accounts[5];
    int totalAccounts = 0;

    int choice;
    do {
        cout << "\n========= Banking System Menu =========\n";
        cout << "1. Create Account\n";
        cout << "2. Deposit\n";
        cout << "3. Withdraw\n";
        cout << "4. Show Balance\n";
        cout << "5. Display Account Info\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        int accNo;
        string name;
        double bal, amount;
        bool found = false;

        switch (choice) {
```

```cpp
case 1:
    if (totalAccounts < 5) {
        cout << "Enter account number: ";
        cin >> accNo;
        cout << "Enter account holder name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter initial balance: ";
        cin >> bal;
        accounts[totalAccounts].setAccountDetails(accNo, name, bal);
        totalAccounts++;
        cout << "Account created successfully!\n";
    } else {
        cout << "Account limit reached (5 accounts max).\n";
    }
    break;

case 2:
    cout << "Enter account number: ";
    cin >> accNo;
    cout << "Enter amount to deposit: ";
    cin >> amount;
    found = false;
    for (int i = 0; i < totalAccounts; i++) {
        if (accounts[i].getAccountNumber() == accNo) {
            accounts[i].deposit(amount);
            found = true;
            break;
        }
    }
    if (!found) cout << "Account not found.\n";
    break;

case 3:
    cout << "Enter account number: ";
    cin >> accNo;
    cout << "Enter amount to withdraw: ";
    cin >> amount;
    found = false;
    for (int i = 0; i < totalAccounts; i++) {
        if (accounts[i].getAccountNumber() == accNo) {
            accounts[i].withdraw(amount);
            found = true;
            break;
        }
    }
    if (!found) cout << "Account not found.\n";
    break;

case 4:
```

```cpp
            cout << "Enter account number: ";
            cin >> accNo;
            found = false;
            for (int i = 0; i < totalAccounts; i++) {
                if (accounts[i].getAccountNumber() == accNo) {
                    cout << "Balance: " << accounts[i].getBalance(accNo) << endl;
                    found = true;
                    break;
                }
            }
            if (!found) cout << "Account not found.\n";
            break;

        case 5:
            cout << "Enter account number: ";
            cin >> accNo;
            found = false;
            for (int i = 0; i < totalAccounts; i++) {
                accounts[i].displayAccountInfo(accNo);
                found = true;
                break;
            }
            if (!found) cout << "Account not found.\n";
            break;

        case 6:
            cout << "Exiting... Thank you for using the banking system!\n";
            break;

        default:
            cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 6);

    return 0;
}
```

SAMPLE OUTPUT:

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 1
Enter account number: 101
Enter account holder name: Pawan Joshi
Enter initial balance: 5000
Account created successfully!

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 2
Enter account number: 101
Enter amount to deposit: 2000
Amount deposited successfully.

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 3
Enter account number: 101
Enter amount to withdraw: 1000
Amount withdrawn successfully.

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 4
Enter account number: 101
Balance: 6000

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 5
Enter account number: 101

Account Number: 101
Account Holder Name: Pawan Joshi
Available Balance: 6000

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 6
Exiting... Thank you for using the banking system!

Q10. Define a class Hotel in C++ with the following specifications:
Private Members
- **Rno – Data member to store room number.**
- **Name – Data member to store customer name.**
- **Tariff – Data member to store charges per day.**
- **NOD – Data member to store number of days of stay.**
- **CALC() – Member function to calculate and return the total amount as:**
  - **Amount = NOD * Tariff**
  - **If Amount > 10000, then Amount = 1.05 * NOD * Tariff (5% extra charge).**

Public Members
- **Checkin() – Function to enter the values of Rno, Name, Tariff, and NOD.**
- **Checkout() – Function to display Rno, Name, Tariff, NOD, and the total amount (calculated using CALC() function).**

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

class Hotel {
private:
    int Rno;
    string Name;
    double Tariff;
    int NOD;

    double CALC() {
        double amount = NOD * Tariff;
        if (amount > 10000) {
            amount = 1.05 * amount;
        }
        return amount;
    }

public:
    void Checkin() {
        cout << "Enter Room Number: ";
        cin >> Rno;
        cin.ignore();
        cout << "Enter Customer Name: ";
        getline(cin, Name);
        cout << "Enter Tariff per day: ";
        cin >> Tariff;
        cout << "Enter Number of Days: ";
        cin >> NOD;
    }

    void Checkout() {
        cout << "\n------ Bill Details ------\n";
        cout << "Room Number   : " << Rno << endl;
```

```cpp
        cout << "Customer Name : " << Name << endl;
        cout << "Tariff per Day: " << Tariff << endl;
        cout << "Days Stayed   : " << NOD << endl;
        cout << "Total Amount  : " << CALC() << endl;
        cout << "-------------------------\n";
    }
};

int main() {
    Hotel h;
    h.Checkin();
    h.Checkout();
    return 0;
}
```

SAMPLE OUTPUT:

Case 1: Amount ≤ 10000

Enter Room Number: 101
Enter Customer Name: Pawan Joshi
Enter Tariff per day: 2000
Enter Number of Days: 4

------ Bill Details ------
Room Number   : 101
Customer Name : Pawan Joshi
Tariff per Day: 2000
Days Stayed   : 4
Total Amount : 8000

Case 2: Amount > 10000 (5% extra applied)

Enter Room Number: 202
Enter Customer Name: Raj Sharma
Enter Tariff per day: 3000
Enter Number of Days: 4

------ Bill Details ------
Room Number   : 202
Customer Name : Raj Sharma
Tariff per Day: 3000
Days Stayed   : 4
Total Amount : 12600

Q11. Imagine a tollbooth with a class called TollBooth. The two data items are of type unsigned int and double to hold the total number of cars and total amount of money collected. A constructor initializes both of these data members to 0. A member function called payingCar( )increments the car total and adds 0.5 to the cash total. Another function called nonPayCar( ) increments the car total but adds nothing to the cash total. Finally a member function called display( )shows the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car and another to count a non paying car. Pushing the ESC key should cause the program to print out the total number of cars and total cash and then exit.

```cpp
#include <iostream>
#include <conio.h>
using namespace std;

class TollBooth{
  private:
    unsigned int totalCars;
    double totalAmount;

    public:
    TollBooth(){
      totalCars=0;
      totalAmount=0;
    }

    void payingCar(){
      totalAmount+=0.5;
      totalCars++;
    }

    void NonPayCar(){
      totalCars++;
    }

    void display(){
      cout<<"Total Cars:"<<totalCars<<endl;
      cout<<"Total Amount:"<<totalAmount<<endl;

    }

};

int main(){

  TollBooth t1;
  char ch;

  while(true){
    cout<<"Enter p for paying , n for non paying car and esc to End"<<endl;
    cout<<endl;
```

```
  ch=_getch();
  if (ch==27){
    t1.display();
    break;
  }

 if (ch=='p' ){
   t1.payingCar();
   cout<<"PAYING CAR ADDED\n"<<endl;
 }
 else if(ch=='n' ){
   t1.NonPayCar();
  cout<<"NON PAYING CAR ADDED\n"<<endl;
 }
 }
}
```

SAMPLE OUTPUT:

Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

n
NON PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

<ESC pressed>
Total Cars:4
Total Amount:1.5

Q12. Create a class called Time that has separate int member data for hours, minutes and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values. A member function should display it in 11:59:59 format. A member function named add() should add two objects of type time passed as arguments. A main ( ) program should create two initialized values together, leaving the result in the third time variable. Finally it should display the value of this third variable.

```cpp
#include <iostream>

using namespace std;

class Time{
 private:
   int hours;
   int minutes;
   int seconds;

 public:
  Time(){
    hours=0;
    minutes=0;
    seconds=0;
  }

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

  void add(Time t11, Time t22){
    hours= t11.hours+ t22.hours;
    minutes= t11.minutes+ t22.minutes;
    seconds= t11.seconds+ t22.seconds;

    if(hours>12){
      hours%=12;
      minutes%=60;
      seconds%=60;
    }
  }

  void display(){
    cout<<hours<<" : "<< minutes <<" : " <<seconds <<endl;
  }

};

int main(){
 Time t1(12,12,12), t2(1, 1, 1),t3;

 t3.add(t1, t2);
```

```
  t3.display();
  return 0;
}
```

SAMPLE OUTPUT:

1 : 13 : 13

Q13. Create class SavingsAccount. Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest() to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12.This interest should be added tosavingsBalance. Provide a static method modifyInterestRate() that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of Rs2000.00 and Rs3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.

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

class SavingsAccount {
private:
    double savingsBalance;
    static double annualInterestRate;

public:
    SavingsAccount(double balance = 0.0) {
        savingsBalance = balance;
    }

    void calculateMonthlyInterest() {
        double monthlyInterest = (savingsBalance * annualInterestRate) / 12;
        savingsBalance += monthlyInterest;
    }

    static void modifyInterestRate(double newRate) {
        annualInterestRate = newRate;
    }

    void displayBalance() const {
        cout << "Balance: Rs " << savingsBalance << endl;
    }
};

double SavingsAccount::annualInterestRate = 0.0;

int main() {
    SavingsAccount saver1(2000.00);
    SavingsAccount saver2(3000.00);

    SavingsAccount::modifyInterestRate(0.04);

    cout << "After setting interest rate to 4% and adding monthly interest:" << endl;
    saver1.calculateMonthlyInterest();
    saver2.calculateMonthlyInterest();
    cout << "Saver1 "; saver1.displayBalance();
    cout << "Saver2 "; saver2.displayBalance();

    cout << endl;
```

```cpp
    SavingsAccount::modifyInterestRate(0.05);

    cout << "After setting interest rate to 5% and adding next month's interest:" << endl;
    saver1.calculateMonthlyInterest();
    saver2.calculateMonthlyInterest();
    cout << "Saver1 "; saver1.displayBalance();
    cout << "Saver2 "; saver2.displayBalance();

    return 0;
}
```

SAMPLE OUTPUT:

After setting interest rate to 4% and adding monthly interest:
Saver1 Balance: Rs 2006.67
Saver2 Balance: Rs 3010

After setting interest rate to 5% and adding next month's interest:
Saver1 Balance: Rs 2015.03
Saver2 Balance: Rs 3022.54