Assignment : 8 Programming

Name:-Jeevan Rajpurohit

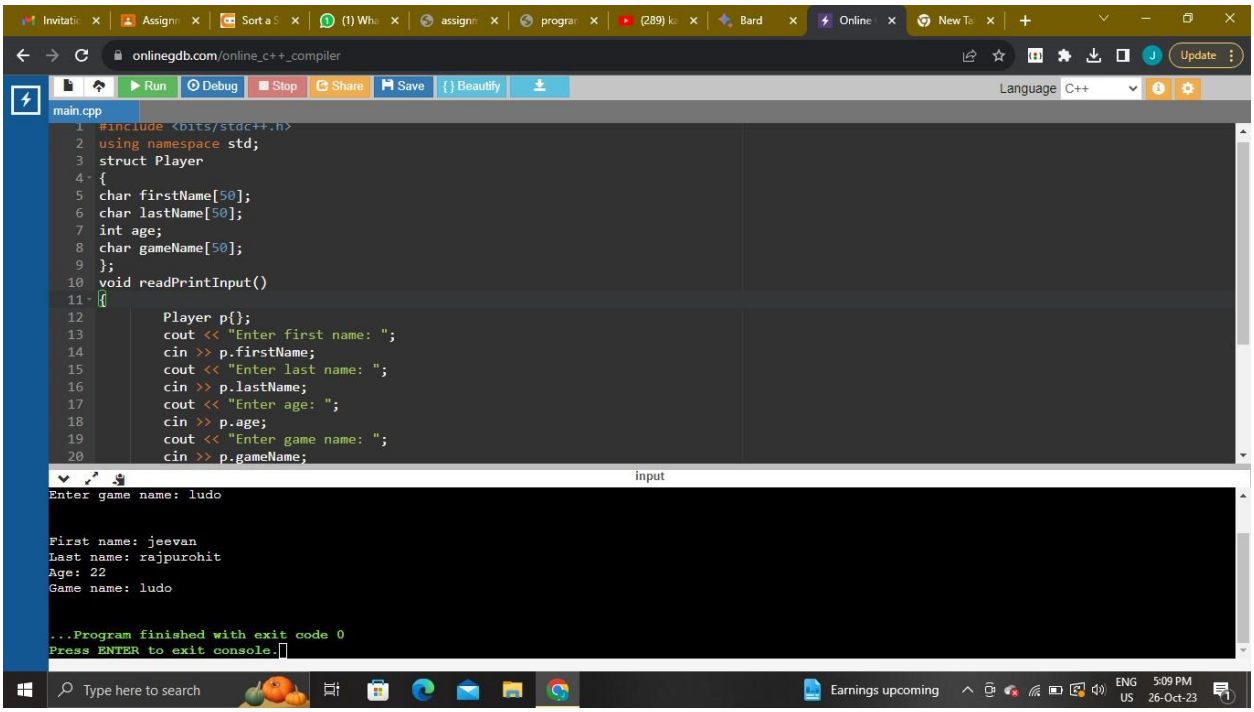
Student id :- 202312090

1). Create a structure representing a player. It should have player's first name, last name, age and name of the game. Do the following activities on this structure-

a). Write a function that creates object of this structure, reads inputs from user for the properties of the object and prints the read values.

```
#include <bits/stdc++.h>

using namespace std;

struct Player

{

char firstName[50];

char lastName[50];

int age;

char gameName[50];

};

void readPrintInput()

{

    Player p{};

    cout << "Enter first name: ";

    cin >> p.firstName;

    cout << "Enter last name: ";

    cin >> p.lastName;

    cout << "Enter age: ";

    cin >> p.age;

    cout << "Enter game name: ";

    cin >> p.gameName;

    cout << endl << endl;
```

```
    cout << "First name: " << p.firstName << endl;
    cout << "Last name: " << p.lastName << endl;
    cout << "Age: " << p.age << endl;
    cout << "Game name: " << p.gameName << endl;
}
int main()
{
  readPrintInput();
}
```

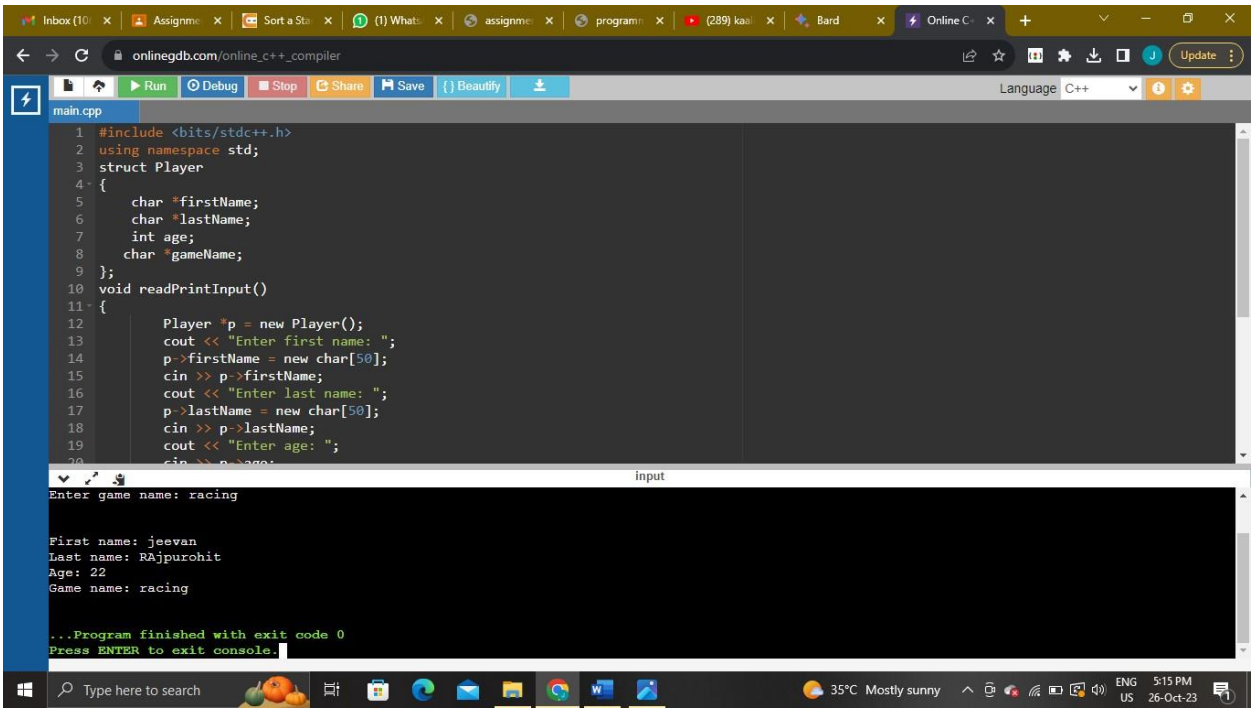


b). Do a. above using pointer to the structure object.

```
#include <bits/stdc++.h>
using namespace std;
struct Player
{
  char *firstName;
  char *lastName;
  int age;
```

```
   char *gameName;
};
void readPrintInput()
{
	Player *p = new Player();
	cout << "Enter first name: ";
	p->firstName = new char[50];
	cin >> p->firstName;
	cout << "Enter last name: ";
	p->lastName = new char[50];
	cin >> p->lastName;
	cout << "Enter age: ";
	cin >> p->age;
	cout << "Enter game name: ";
	p->gameName = new char[50];
	cin >> p->gameName;
	cout << endl
	<< endl;
	cout << "First name: " << p->firstName << endl;
	cout << "Last name: " << p->lastName << endl;
	cout << "Age: " << p->age << endl;
	cout << "Game name: " << p->gameName << endl;
}
int main()
{
readPrintInput();
```

}

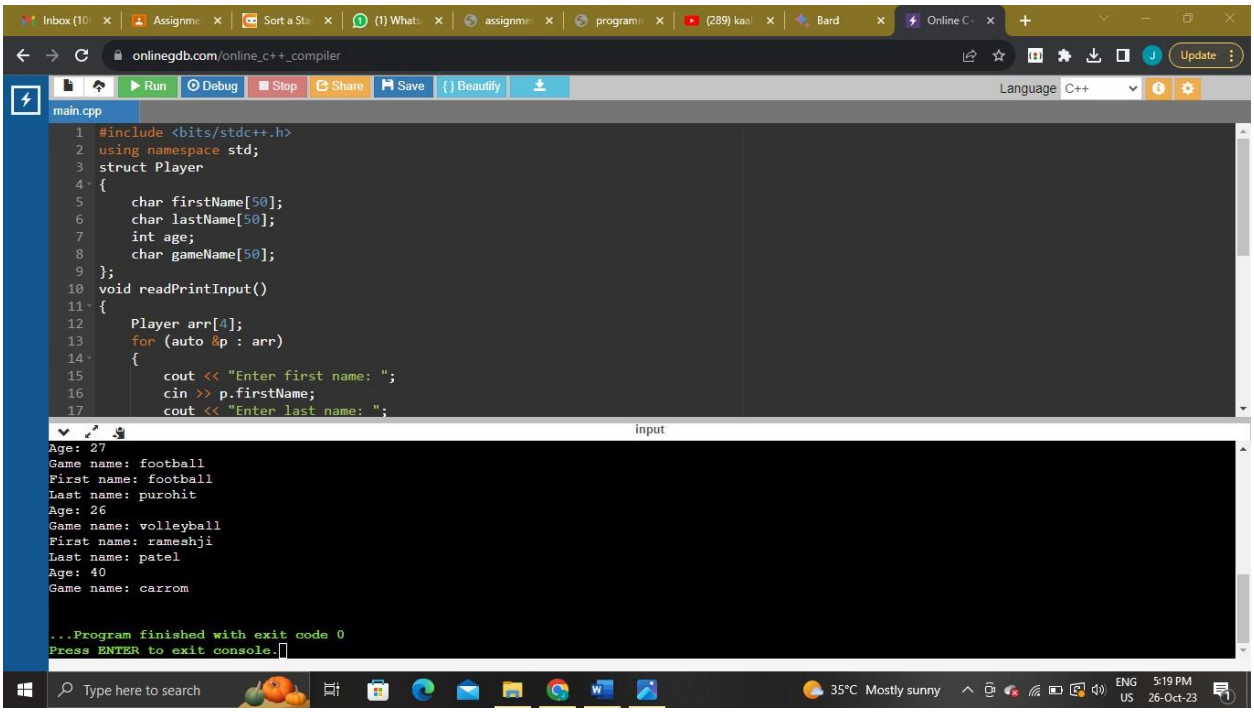


c). Create an array of size 4 and read from user the values and print them

```
#include <bits/stdc++.h>

using namespace std;

struct Player

{

   char firstName[50];

   char lastName[50];

   int age;

   char gameName[50];

};

void readPrintInput()

{

   Player arr[4];

   for (auto &p : arr)

   {

      cout << "Enter first name: ";
```

```
        cin >> p.firstName;
        cout << "Enter last name: ";
        cin >> p.lastName;
        cout << "Enter age: ";
        cin >> p.age;
        cout << "Enter game name: ";
        cin >> p.gameName;
}
  cout << endl
<< endl;
for (auto &p : arr)
{
    cout << "First name: " << p.firstName << endl;
    cout << "Last name: " << p.lastName << endl;
    cout << "Age: " << p.age << endl;
    cout << "Game name: " << p.gameName << endl;
}
}
int main()
{
readPrintInput();
```

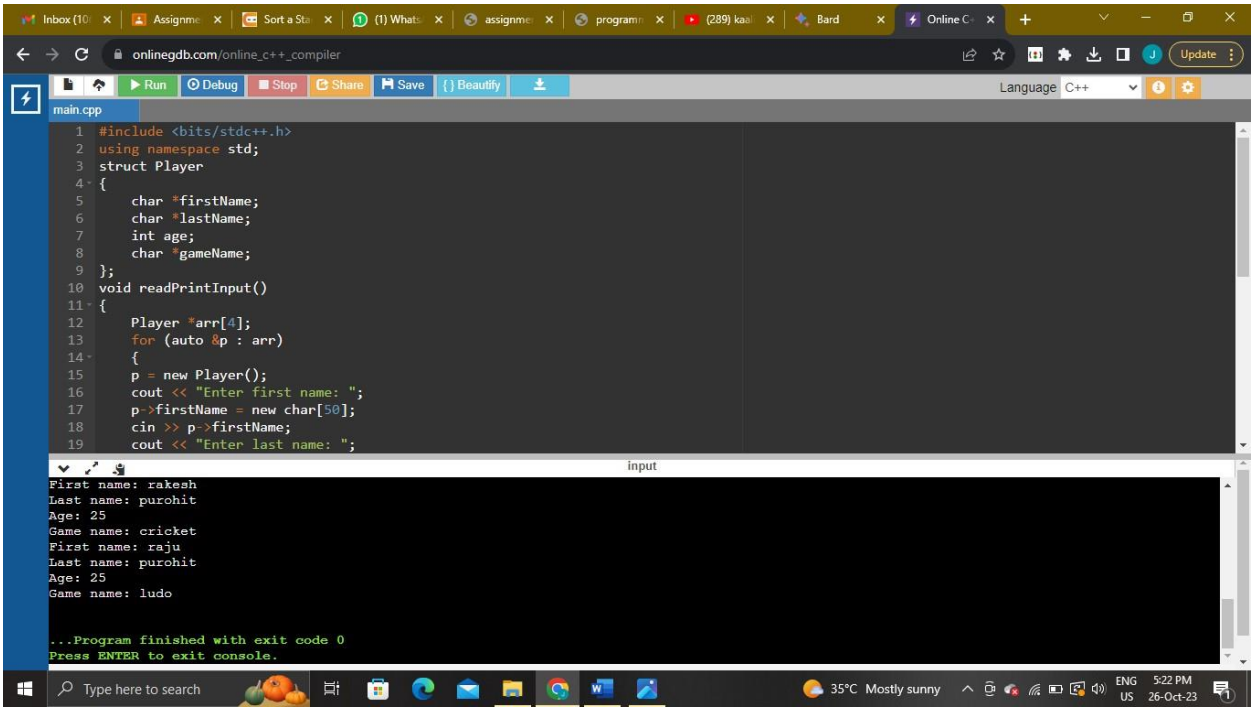
}



d).

```
#include <bits/stdc++.h>

using namespace std;

struct Player

{

   char *firstName;

   char *lastName;

   int age;

   char *gameName;

};

void readPrintInput()

{

   Player *arr[4];

   for (auto &p : arr)

   {

   p = new Player();
```

```
    cout << "Enter first name: ";
    p->firstName = new char[50];
    cin >> p->firstName;
    cout << "Enter last name: ";
    p->lastName = new char[50];
    cin >> p->lastName;
    cout << "Enter age: ";
    cin >> p->age;
    cout << "Enter game name: ";
    p->gameName = new char[50];
    cin >> p->gameName;
    }
    cout << endl
    << endl;
for (auto p : arr)
{
    cout << "First name: " << p->firstName << endl;
    cout << "Last name: " << p->lastName << endl;
    cout << "Age: " << p->age << endl;
    cout << "Game name: " << p->gameName << endl;
}
}
int main()
{
readPrintInput();
}
```

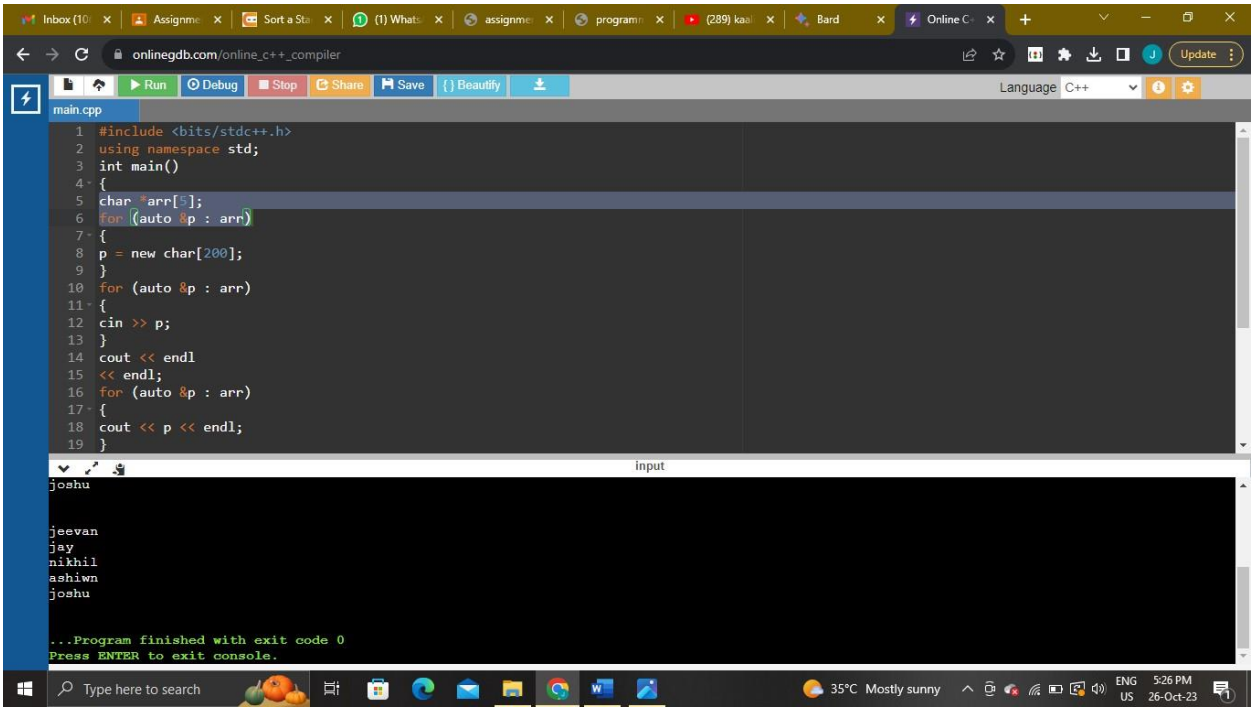


2).

Write a program that declares array of 5 pointers to strings. That is, each element of this 5 element array is a char*. Allocate memory to read a maximum of 200 characters for each element of the array. Read 5 names from the user using these pointers. Display all names. Now, re-allocate memory to these pointers equal to the size of the names. Display the names again to verify it is correct allocation. Don't forget to free the pointers at end of the program.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

char *arr[5];

for (auto &p : arr)

{

p = new char[200];

}

for (auto &p : arr)
```

```
{
cin >> p;
}
cout << endl
<< endl;
for (auto &p : arr)
{
cout << p << endl;
}
cout << endl
<< endl;
for (auto &p : arr)
{
char *temp_arr = new char[strlen(p) + 1];
memcpy(temp_arr, p, strlen(p) + 1);
delete[] p;
p = temp_arr;
cout << temp_arr << endl;
}
}
```

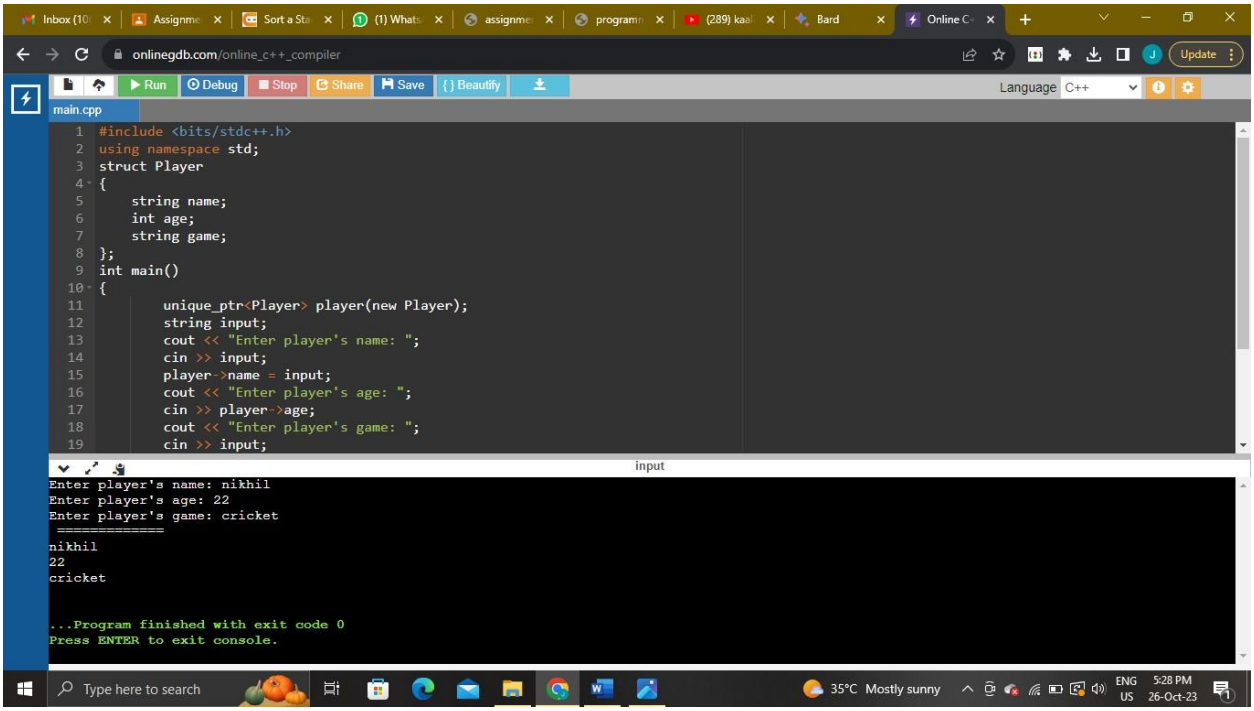


3).

Declare a structure Player. It has following member variables (properties):
name: a pointer to string that is name of the player
age: an integer to store name
game: a pointer to string that stores game of the player
The program should read string inputs from user in single pointer variable named input. This input variable is pointer whose memory is allocated on heap with size of 100 characters. After user given an input (eg. for name) the program copies this string to the field of the structure (eg. the name field). Make sure only required characters are copied.
Display the name, age and game of the player after reading the data. Make sure you de-allocate all memories allocated before program exits.

```
#include <bits/stdc++.h>

using namespace std;

struct Player

{

   string name;

   int age;

   string game;

};
```

```
int main()
{
	unique_ptr<Player> player(new Player);
	string input;
	cout << "Enter player's name: ";
	cin >> input;
	player->name = input;
	cout << "Enter player's age: ";
	cin >> player->age;
	cout << "Enter player's game: ";
	cin >> input;
	player->game = input;
	cout << " ============== " << endl;
	cout << player->name << endl;
	cout << player->age << endl;
	cout << player->game << endl;
return 0;
}
```



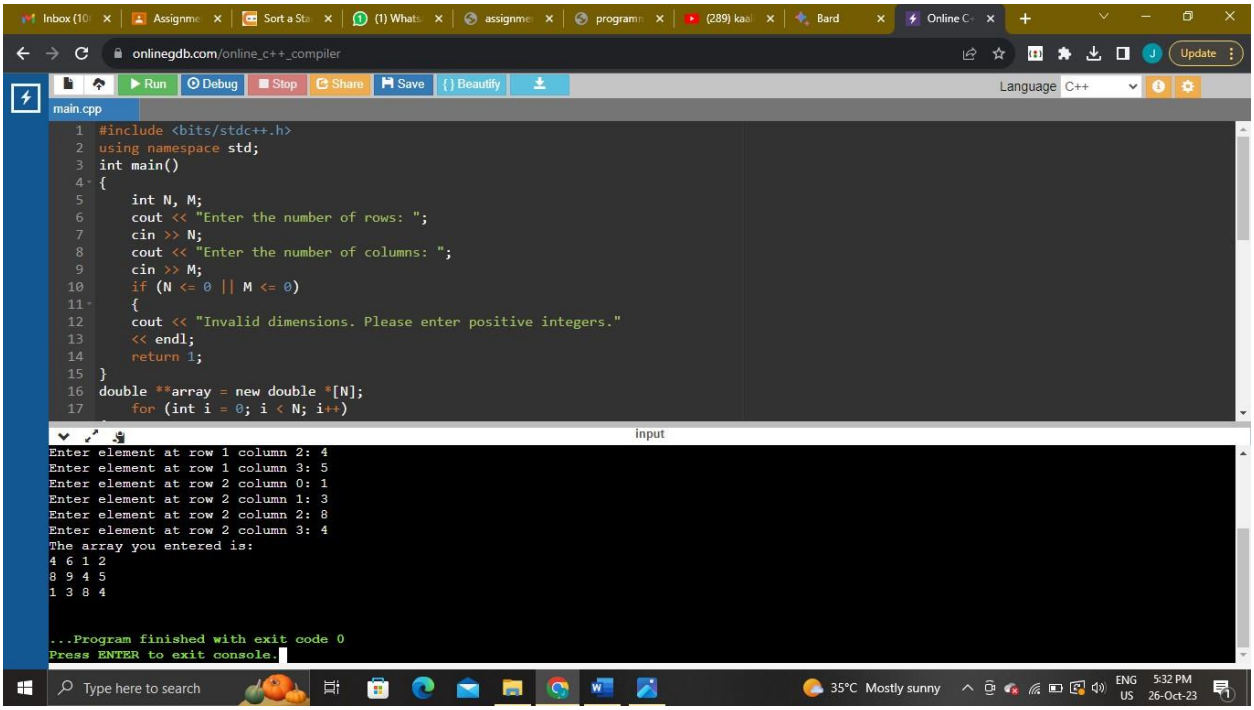
4).

Write a program that reads the NxM dimensions of a two-dimensional array of doubles. If user enters 0 or less for either N or M, the program exits. After reading the size of array from user, i.e. N for rows and M for columns, it allocates memory as required. Then, it reads the numbers from user and puts these in appropriate 'cells' for the matrix. It then displays the numbers read and free's up memory before exiting.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

   int N, M;

   cout << "Enter the number of rows: ";

   cin >> N;

   cout << "Enter the number of columns: ";

   cin >> M;

   if (N <= 0 || M <= 0)

   {
```

```
    cout << "Invalid dimensions. Please enter positive integers."
    << endl;
    return 1;
}
double **array = new double *[N];
    for (int i = 0; i < N; i++)
{
        array[i] = new double[M];
}
for (int i = 0; i < N; i++)
{
     for (int j = 0; j < M; j++)
     {
          cout << "Enter element at row " << i << " column " << j <<": ";
          cin >> array[i][j];
     }
}
cout << "The array you entered is:" << endl;
for (int i = 0; i < N; i++)
{
        for (int j = 0; j < M; j++)
        {
              cout << array[i][j] << " ";
        }
        cout << endl;
}
for (int i = 0; i < N; i++)
{
        delete[] array[i];
```

```
    }
    delete[] array;
    return 0;
}
```



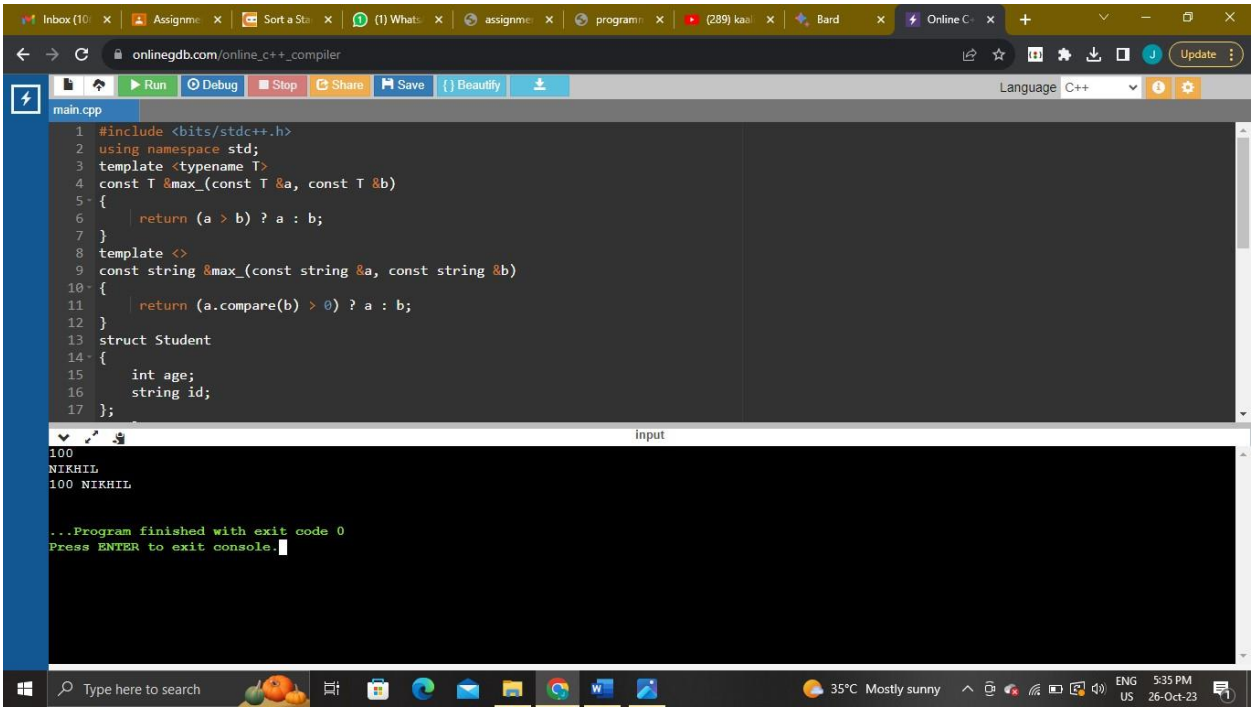
5).

Create a max function template and use it for int, std::string, and Student type with age and ID as members.

```
#include <bits/stdc++.h>
using namespace std;
template <typename T>
const T &max_(const T &a, const T &b)
{
    return (a > b) ? a : b;
}
template <>
const string &max_(const string &a, const string &b)
{
```

```
    return (a.compare(b) > 0) ? a : b;
}
struct Student
{
   int age;
   string id;
};
template <>
const Student &max_(const Student &a, const Student &b)
{
   if (a.age == b.age)
{
      return (max_(a.id, b.id) == a.id) ? a : b;
}
      return (max_(a.age, b.age) == a.age) ? a : b;
}
int main()
{
      int x, y;
      x = 99, y = 100;
      cout << max_(x, y) << endl;
      string str1 = "JEEVAN";
      string str2 = "NIKHIL";
      cout << max_(str1, str2) << endl;
      Student s1, s2;
      s1.age = x;
      s2.age = y;
      s1.id = str1;
      s2.id = str2;
```

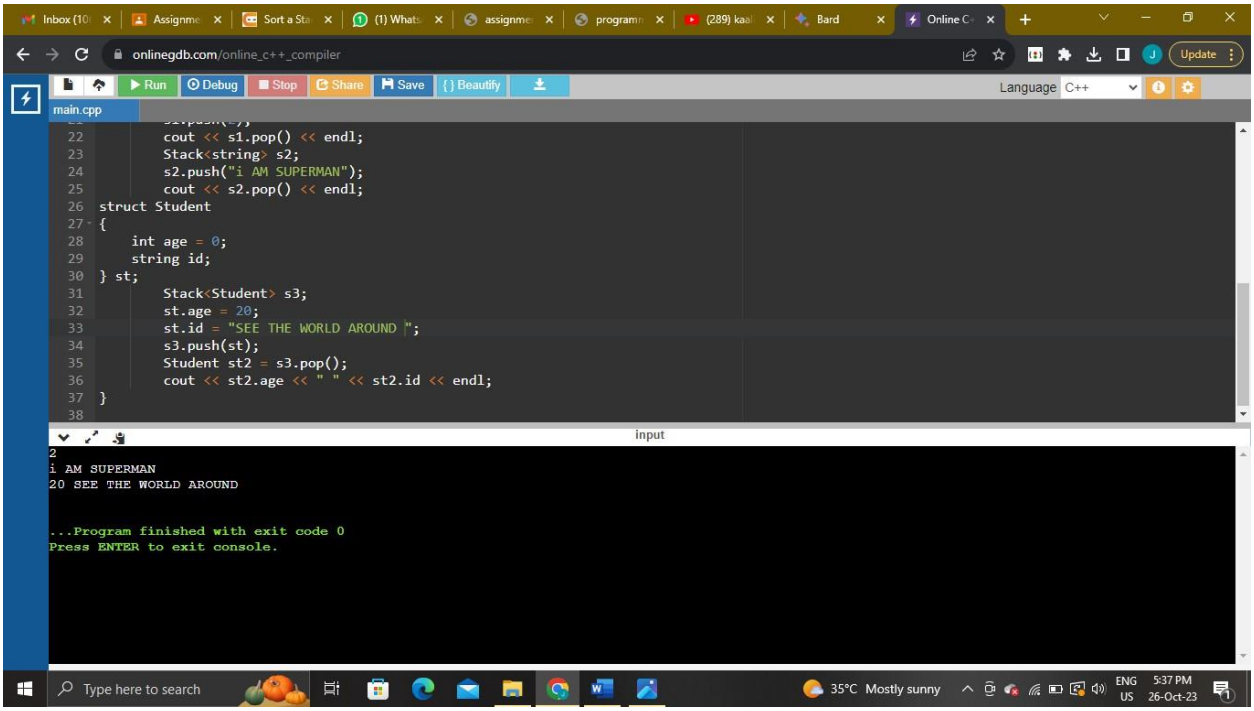
```
    Student max_s = max_(s1, s2);
    cout << max_s.age << " " << max_s.id << endl;
}
```



6). Build a stack class using templates and use it for int, std::string, and Student type with age and ID as members.

```
#include <bits/stdc++.h>
using namespace std;
template <typename T>
struct Stack
{
   int top = 0;
   T elements[5];
void push(const T &element)
{
   elements[top++] = element;
}
   T pop()
```

```
{
   T element = elements[--top];
return element;
}
};
int main()
{
      Stack<int> s1;
      s1.push(2);
      cout << s1.pop() << endl;
      Stack<string> s2;
      s2.push("i AM SUPERMAN");
      cout << s2.pop() << endl;
struct Student
{
   int age = 0;
   string id;
} st;
      Stack<Student> s3;
      st.age = 20;
      st.id = "SEE THE WORLD AROUND ";
      s3.push(st);
      Student st2 = s3.pop();
      cout << st2.age << " " << st2.id << endl;
}
```



Q7 and Q8 were removed by SIR.