FAT 1 - CSE1002

UNDERWEAR EDITION

Count Vertical Lines Connecting Horizontal Lines

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
    vector<pair<pair<int, int>, pair<int, int>>> vert, hor;
    pair<pair<int, int>, pair<int, int>> t;
    int n; cin >> n;
    for(int i = 0; i < n; i++){
        cin >> t.first.first >> t.first.second;
        cin >> t.second.first >> t.second.second;
        if(t.first.first == t.second.first) vert.push_back(t);
        if(t.second.second == t.first.second) hor.push_back(t);
    }
    int ans = 0;
    for(auto &i : vert){
        int cnt = 0;
        for(auto &j : hor){
            if(i.first == j.first || i.first == j.second) cnt++;
            else if(i.second == j.first || i.second == j.second) cnt++;
        }
        if(cnt >= 2) ans++;
    }
    cout << ans << endl;
    return 0;
}
```

3D POINTS INCREMENT

```
istream& operator>>(istream& in , cartesian& car){
   in >> car.xpos;
   in >> car.ypos;
   in >> car.zpos;
   return in;
}

ostream& operator << (ostream& out, cartesian& car){
   out << car.xpos << endl;
   out << car.ypos << endl;
   out << car.zpos << endl;
   return out;
}

cartesian cartesian :: operator + (cartesian& c2){
   cartesian temp;
   temp.xpos = xpos + c2.xpos;
   temp.ypos = ypos + c2.ypos;
   temp.zpos = zpos + c2.zpos;
   return temp;
}

int cartesian:: operator [](int i){
   if(i == 0){
      return xpos ;

   }
   else if(i ==1){
      return ypos;
   }
   else if(i == 2){
      return zpos;
   }
   else {
      cout << "integer exception";
   }
}
```

Given endpoints of 'n' lines, design an algorithm and write a C++ code to find out the number of vertical lines connecting either end of two horizontal lines. Assume that no two horizontal lines have same 'x' and no two vertical lines have same 'y' and all the values given are positive. For example, when eight lines with end points as follows are given as input:

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

int main()
{
    int n;
    cin>>n;
    //horizental lines
    vector<vector<pair<int,int>>> horizontal;
     vector<vector<pair<int,int>>>vertical;
    for(int i=0;i<n;i++){
      int x,y;
      vector<pair<int,int>>v1;
      cin>>x>>y;
      v1.push_back(make_pair(x,y));
      horizontal.push_back(v1);
      cin>>x>>y;
      v1.push_back(make_pair(x,y));
      vertical.push_back(v1);
    }

    int c = 0;
    for(int i = 0; i < vertical.size(); i++)
    {
        pair<int,int> v1, v2;     //end points of vertical line i
        v1 = vertical[i][0];
        //vertical vector
        v2 = vertical[i][1];
        //horizontal vector
        int flag1 = 0;
        int flag2 = 0;
        //second loop
        for(int j = 0; j < horizontal.size(); j++)
        {
            pair<int,int> h1, h2;     //end points of horizental line j
            h1 = horizontal[j][0];
            h2 = horizontal[j][1];
            //if condition
            if(v1 == h1 || v1 == h2)
                flag1 = 1;
            //second if
            if(v2 == h1 || v2 == h2)
                flag2 = 1;
        }
        //third if
        if(flag1 == 1 && flag2 == 1)
            c++;
            //check variable
    }
    cout<<c;
    return 0;
}
```

Given a point in three-dimensional space with x-coordinate, y-coordinate, z-coordinates, Provide a function increment with one and three parameters to increment the values of all the three coordinates. When the function with one parameter is called increment all the three , by same value . When the function with three arguments is called, increment the current co-ordinates: x-coordinate by the value of first parameter, y-coordinate by the value of the second parameter and z-coordinate by the value of the third parameter.

```
#include< iostream >
using namespace std;
class dim
{
     int x,y,z;
     public :
     void get()
     {
     cin>>x>>y>>z;
     }
     void increment(int val)
```

```
        {
        x+=val;
        y+=val;
        z+=val;
        }
        void increment(int val1,int val2)
        {
        x*=val1+val2;
        y*=val1+val2;
        z*=val1+val2;
        }
        void increment(int val1,int val2,int val3)
        {
        x+=val1;
        y+=val2;
        z+=val3;
        }
        void print()
        {
        cout<<x<<"\n"<<y<<"\n"<<z<<"\n";
        }
};
```

```
main()

{

dim d1;

d1.get();

int inc;

cin>>inc;

int p1,p2;

cin>>p1>>p2;
int ix,iy,iz;
cin>>ix>>iy>>iz;
d1.increment(inc);
d1.print();
d1.increment(p1,p2);
d1.print();
d1.increment(ix,iy,iz);
d1.print();
}
```

FORM RECTANGLE TRIANGLE

```
// C++ code to count the number of
// possible triangles using brute
// force approach
#include <bits/stdc++.h>
using namespace std;

// Function to count all possible
// triangles with arr[] elements
int findNumberOfTriangles(int arr[], int n)
{
     // Count of triangles
     int count = 0;

     // The three loops select three
     // different values from array
     for (int i = 0; i < n; i++) {
          for (int j = i + 1; j < n; j++) {

               // The innermost loop checks for
               // the triangle property
               for (int k = j + 1; k < n; k++)

                    // Sum of two sides is greater
                    // than the third
                    if (
                         arr[i] + arr[j] > arr[k]
                         && arr[i] + arr[k] > arr[j]
                         && arr[k] + arr[j] > arr[i])
                         count++;
          }
     }
     return count;
}
// Driver code
int main()
{
     int arr[] = { 10, 21, 22, 100, 101, 200, 300 };
     int size = sizeof(arr) / sizeof(arr[0]);

     cout
          << "Total number of triangles possible is "
          << findNumberOfTriangles(arr, size);

     return 0;
}
```

Construct Triangle with Maximum Area

A triangle can be formed with three points. Given ‘n’ points, we can find the triplets which form a triangle. So, given ‘n’ points, we can form triangles whose vertices are among the given ‘n’ points. Of these triangles, we call the triangle with maximum area as ‘Maximum area triangle’. Given 'n' points, design an algorithm and write a C++ code to determine the ‘maximum area triangle’ that can be formed with the given points. Print the area of the ‘maximum area triangle’ formed and their vertices in an ascending order. If more than one ‘maximum area triangle’ exists then print them in an ascending order as per their vertices.

Code 1: -

```
#include<iostream>

using namespace std;

int main()

{

    int a1,b1,c1, c=0, temp1;

    double temp, maxar=0;



    int n;

    cin>>n;



    int x[n][2];

    int cod[n][3][2];



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

```
    cin>>x[i][0]>>x[i][1];
}

for(int i=0; i<n-1; i++){
    for(int j=0; j<n-i-1; j++){
        if(x[j][1]>x[j+1][1]){
            temp1 = x[j][1];
            x[j][1] = x[j+1][1];
            x[j+1][1] = temp1;

            temp1 = x[j][0];
            x[j][0] = x[j+1][0];
            x[j+1][0] = temp1;
        }
    }
}

for(int i=0; i<n-1; i++){
    for (int j=0; j<n-i-1; j++){
        if(x[j][0]>x[j+1][0]){
            temp1 = x[j][0];
            x[j][0] = x[j+1][0];
            x[j+1][0] = temp1;

            temp1 = x[j][1];
            x[j][1]= x[j+1][1];
            x[j+1][1] = temp1;
        }
    }
}
for(int i=0;i<n-2;i++){
    for(int j=i+1;j<n-1;j++){
        for(int k=j+1;k<n;k++){
            a1=x[i][0]*x[j][1]-x[j][0]*x[i][1];
            b1=x[j][0]*x[k][1]-x[k][0]*x[j][1];
            c1=x[k][0]*x[i][1]-x[i][0]*x[k][1];
            temp = abs((a1+b1+c1)/2.0);

            if(maxar<temp){
                maxar=temp;
                c=1;
                cod[c][0][0]=x[i][0];
                cod[c][0][1]=x[i][1];
                cod[c][1][0]=x[j][0];
                cod[c][1][1]=x[j][1];
                cod[c][2][0]=x[k][0];
                cod[c][2][1]=x[k][1];
            }
            else if(maxar==temp){
                c++;
                cod[c][0][0]=x[i][0];
                cod[c][0][1]=x[i][1];
                cod[c][1][0]=x[j][0];
                cod[c][1][1]=x[j][1];
                cod[c][2][0]=x[k][0];
                cod[c][2][1]=x[k][1];
            }
        }
    }
}
```

```
    cout<<maxar<<endl;
    for(int j=1; j<=c; j++){

        for(int i=0; i<3; i++){
            cout<<cod[j][i][0]<<"\t"<<cod[j][i][1]<<endl;
        }
    }

}
```

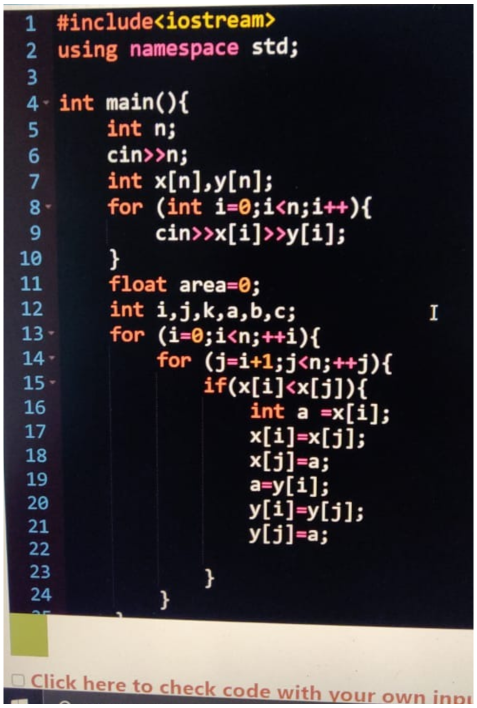
Code 2: -

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

int main()
{
    int n;
    cin>>n;
    int x[n],y[n];
    for (int i=0;i<n;i++)
    {
```

```
    cin>>x[i]>>y[i];

}

float area=0;

int i,j,k,a,b,c;

for(i=0;i<n;++i)

{

    for(j=i+1;j<n;++j)

    {

        if(x[i]<x[j])

        {

            int a=x[i];

            x[i]=x[j];

            x[j]=a;

            a[y]=i;y[i]=y[j];

            y[j]=a;

        }

    }

}

for(i=0;i<n;i++)

{
```

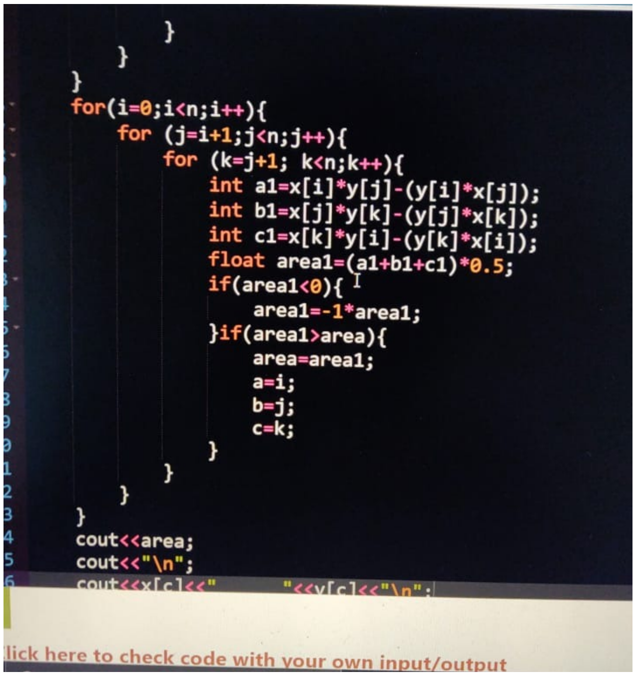
```
for(j=i+1;j<n;j++)

{

    for(k=j+1;k<n;k++)

    {

        int a1=x[i]*y[j]-(y[i]*x[j]);

        int b1=x[j]*y[k]-(y[j]*x[k]);

        int c1=x[k]*y[i]-(y[k]*x[i]);

        float area1=(a1+b1+c1)*0.5;

        if (area1<0)

        {

            area1=-1*area1;

        }

        if(area1>area)

        {

            area=area1;

            a=i;

            b=j;

            c=k;

        }

    }
```

```
        }

    }

    cout<<area;

    cout<<"\n";

    cout<<x[c]<<"       "<<y[c]<<"\n";

    cout<<x[b]<<"       "<<y[b]<<"\n";

    cout<<x[a]<<"       "<<y[a]<<"\n";

}
```



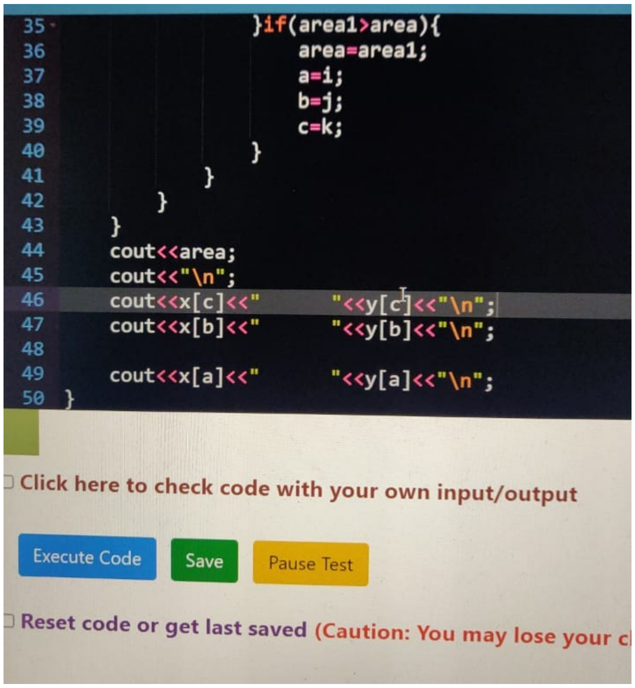
```cpp
#include<iostream>
using namespace std;

int main(){
    int n;
    cin>>n;
    int x[n],y[n];
    for (int i=0;i<n;i++){
        cin>>x[i]>>y[i];
    }
    float area=0;
    int i,j,k,a,b,c;
    for (i=0;i<n;++i){
        for (j=i+1;j<n;++j){
            if(x[i]<x[j]){
                int a =x[i];
                x[i]=x[j];
                x[j]=a;
                a=y[i];
                y[i]=y[j];
                y[j]=a;

            }
        }
```

□ Click here to check code with your own inpu



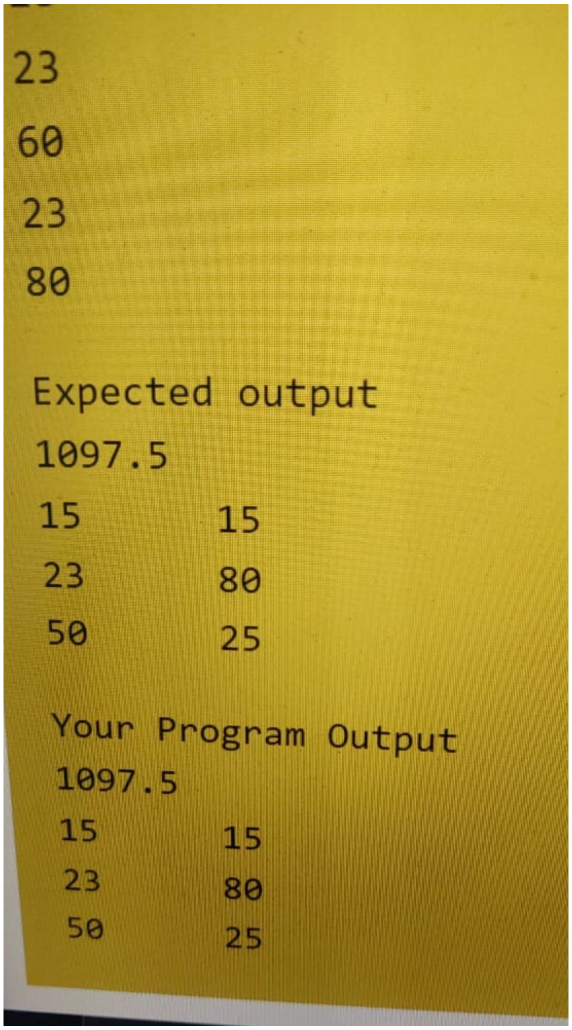
```cpp
                }
            }
        }
    for(i=0;i<n;i++){
        for (j=i+1;j<n;j++){
            for (k=j+1; k<n;k++){
                int a1=x[i]*y[j]-(y[i]*x[j]);
                int b1=x[j]*y[k]-(y[j]*x[k]);
                int c1=x[k]*y[i]-(y[k]*x[i]);
                float area1=(a1+b1+c1)*0.5;
                if(area1<0){
                    area1=-1*area1;
                }if(area1>area){
                    area=area1;
                    a=i;
                    b=j;
                    c=k;
                }
            }
        }
    }
    cout<<area;
    cout<<"\n";
    cout<<x[c]<<"         "<<y[c]<<"\n";
```

lick here to check code with your own input/output



```
                }if(area1>area){
                    area=area1;
                    a=i;
                    b=j;
                    c=k;
                }
            }
        }
    }
    cout<<area;
    cout<<"\n";
    cout<<x[c]<<"       "<<y[c]<<"\n";
    cout<<x[b]<<"       "<<y[b]<<"\n";

    cout<<x[a]<<"       "<<y[a]<<"\n";
}
```

□ Click here to check code with your own input/output
Execute Code
Save
Pause Test
□ Reset code or get last saved (Caution: You may lose your c



```
23
60
23
80

Expected output
1097.5
15         15
23         80
50         25

Your Program Output
1097.5
15         15
23         80
50         25
```

We’ll get some marks at least for this one.