Design and Analysis of Algorithms Lab Assessment-4

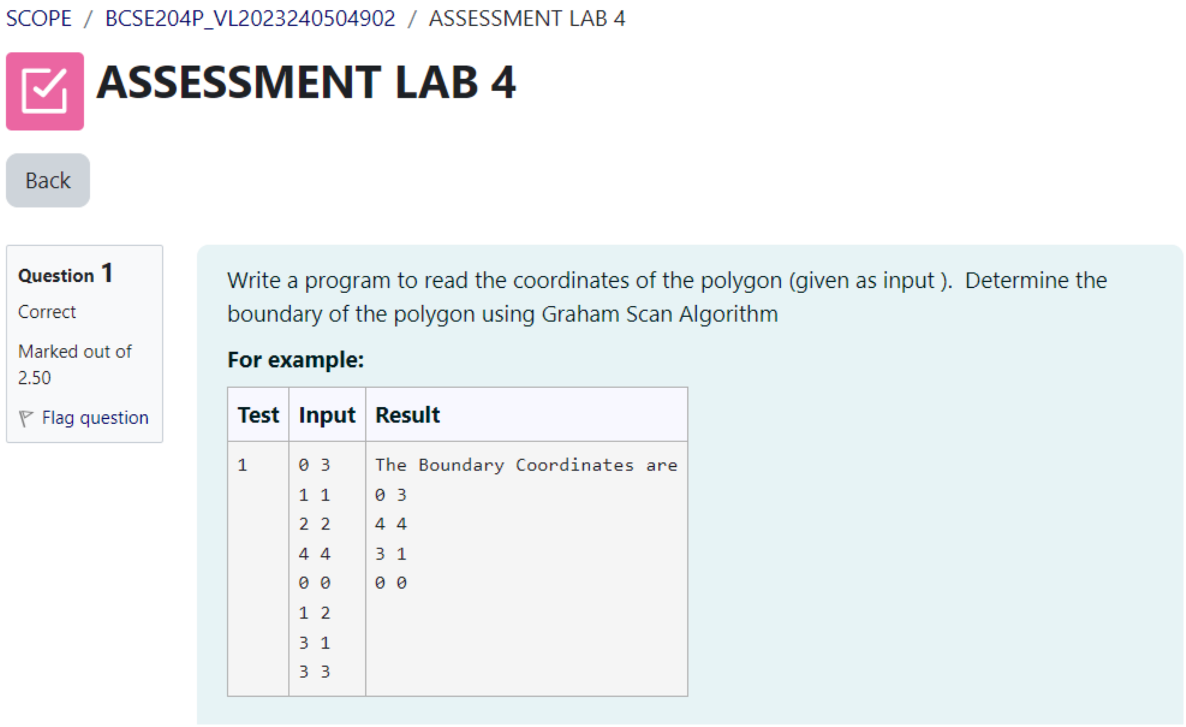
Name: Armaan Indani

Reg. No.: 22BCE3347

Slot : L41+42

Faculty: Dr. Joshva Devdas T – 16700

Question 1:

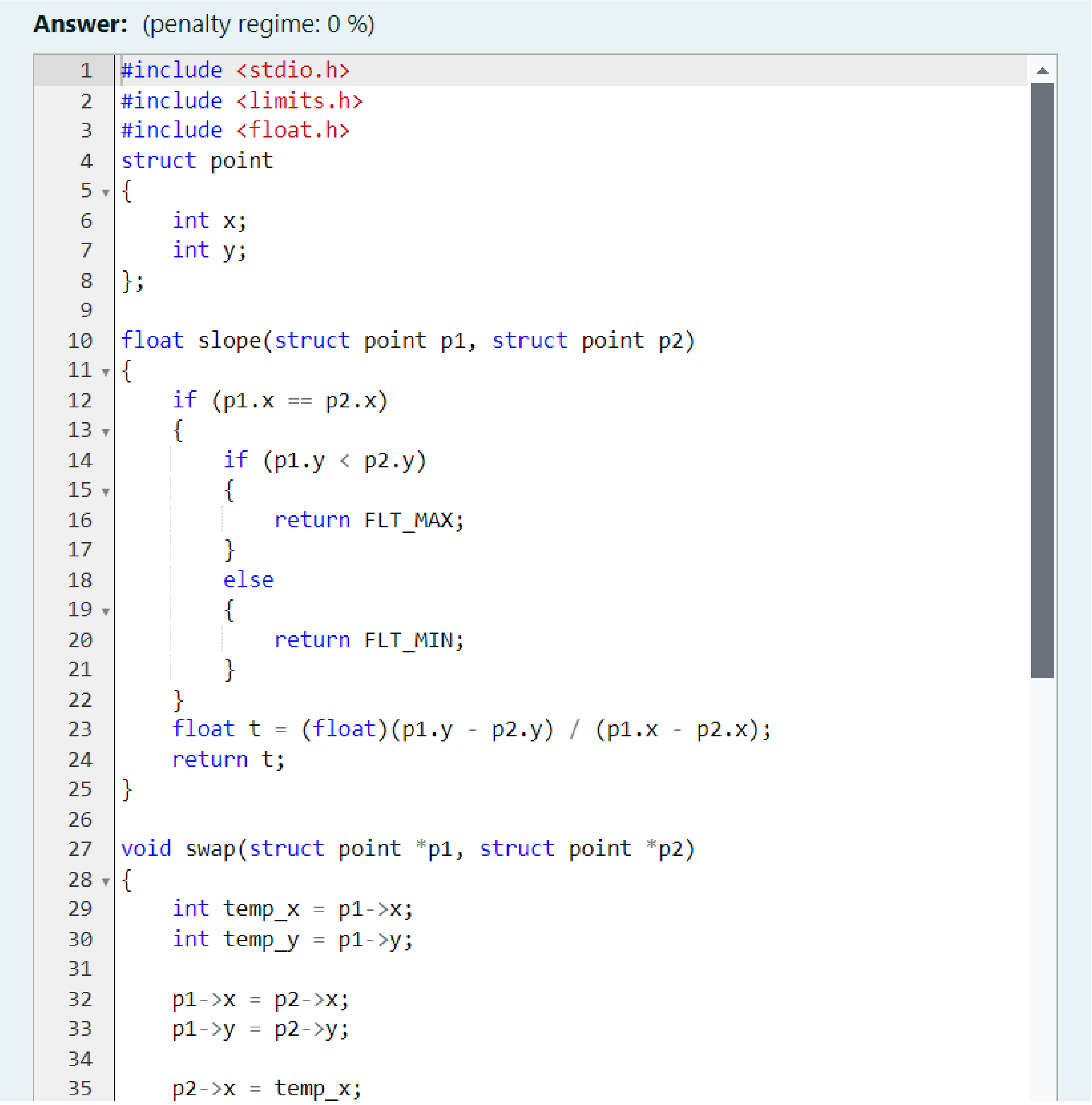


Write a program to read the coordinates of the polygon (given as input). Determine the boundary of the polygon using Graham Scan Algorithm

For example:

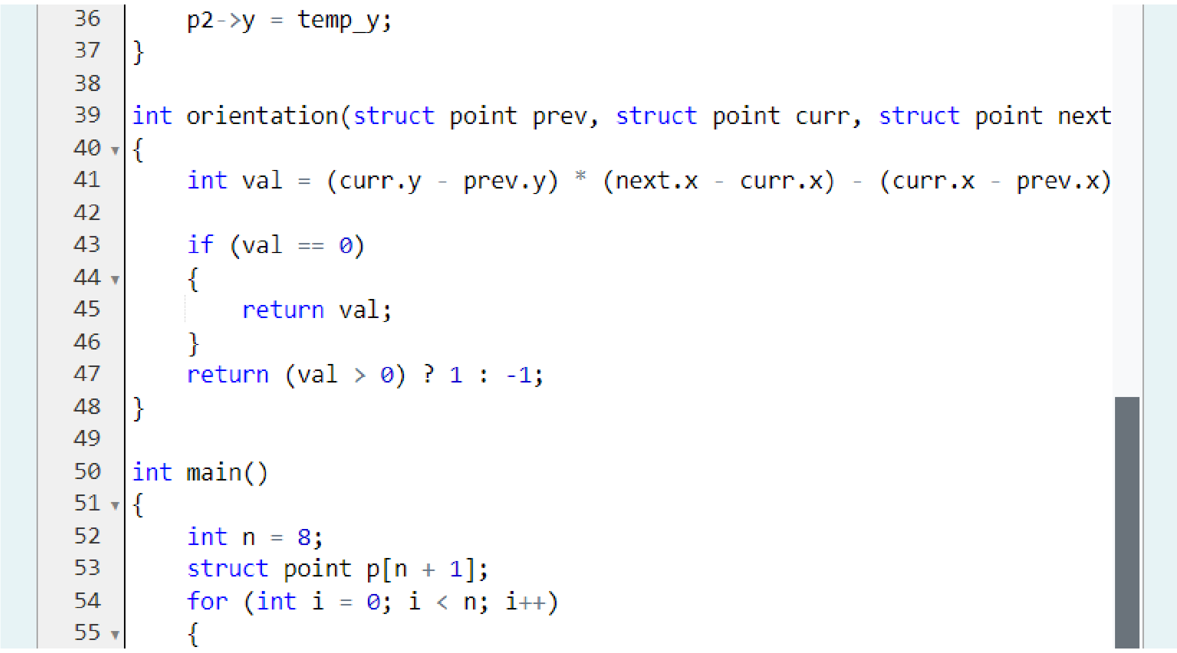
Test	Input	Result

1	0 3
	1 1
	2 2
	4 4
	0 0
	1 2
	3 1
	3 3
		The Boundary Coordinates are
		0 3
		4 4
		3 1
		0 0

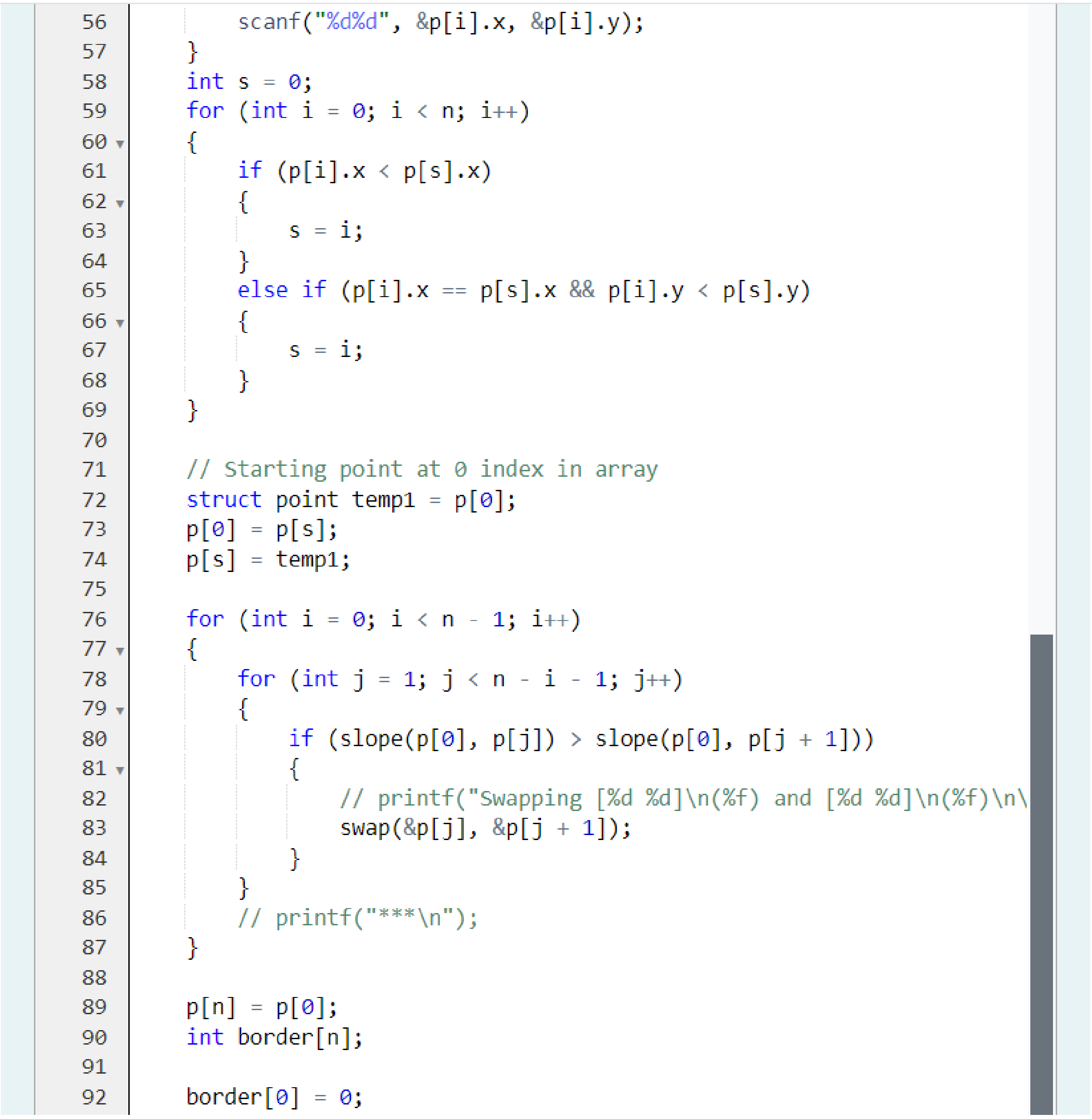


Answer: (penalty regime: 0 %)

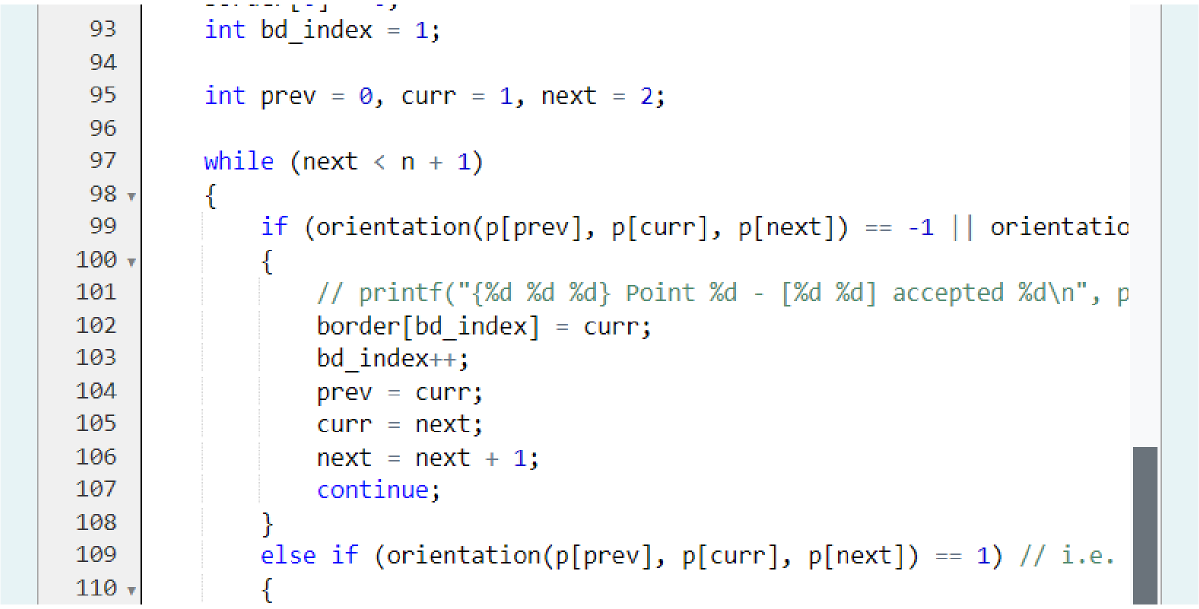
```c
#include <stdio.h>
#include <limits.h>
#include <float.h>
struct point
{
    int x;
    int y;
};

float slope(struct point p1, struct point p2)
{
    if (p1.x == p2.x)
    {
        if (p1.y < p2.y)
        {
            return FLT_MAX;
        }
        else
        {
            return FLT_MIN;
        }
    }
    float t = (float)(p1.y - p2.y) / (p1.x - p2.x);
    return t;
}

void swap(struct point *p1, struct point *p2)
{
    int temp_x = p1->x;
    int temp_y = p1->y;

    p1->x = p2->x;
    p1->y = p2->y;

    p2->x = temp_x;
```



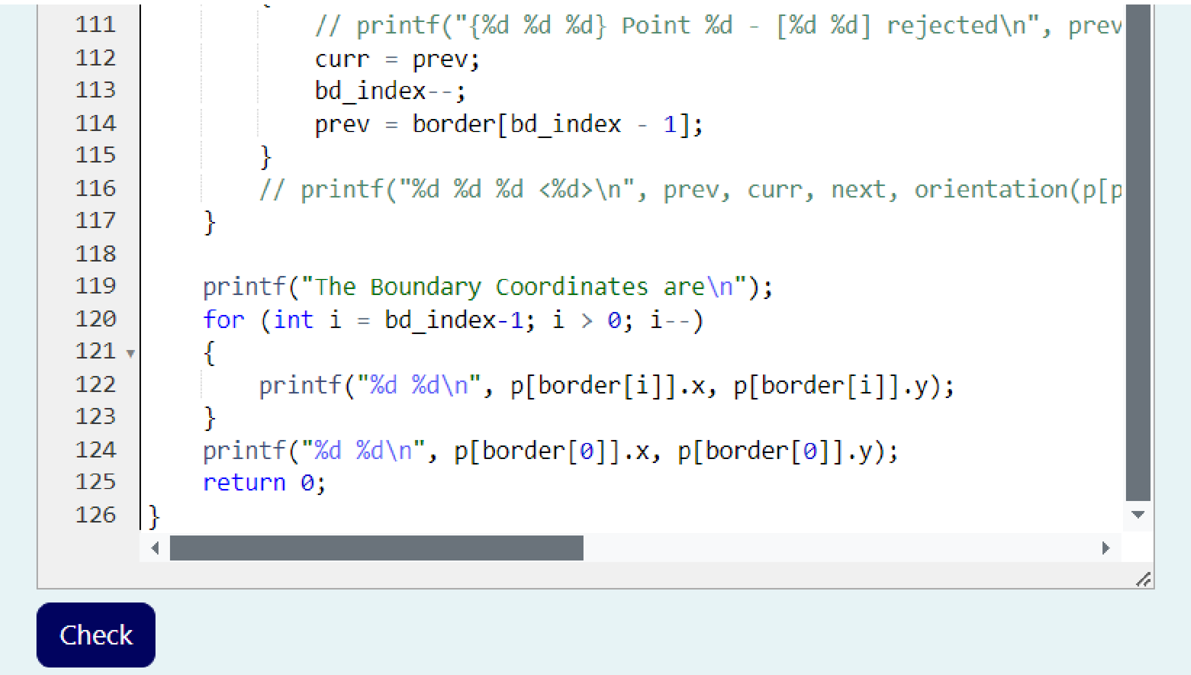
```c
    p2->y = temp_y;
}

int orientation(struct point prev, struct point curr, struct point next
{
    int val = (curr.y - prev.y) * (next.x - curr.x) - (curr.x - prev.x)

    if (val == 0)
    {
        return val;
    }
    return (val > 0) ? 1 : -1;
}

int main()
{
    int n = 8;
    struct point p[n + 1];
    for (int i = 0; i < n; i++)
    {
```



```
        scanf("%d%d", &p[i].x, &p[i].y);
    }
    int s = 0;
    for (int i = 0; i < n; i++)
    {
        if (p[i].x < p[s].x)
        {
            s = i;
        }
        else if (p[i].x == p[s].x && p[i].y < p[s].y)
        {
            s = i;
        }
    }

    // Starting point at 0 index in array
    struct point temp1 = p[0];
    p[0] = p[s];
    p[s] = temp1;

    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 1; j < n - i - 1; j++)
        {
            if (slope(p[0], p[j]) > slope(p[0], p[j + 1]))
            {
                // printf("Swapping [%d %d]\n(%f) and [%d %d]\n(%f)\n\
                swap(&p[j], &p[j + 1]);
            }
        }
        // printf("***\n");
    }

    p[n] = p[0];
    int border[n];

    border[0] = 0;
```

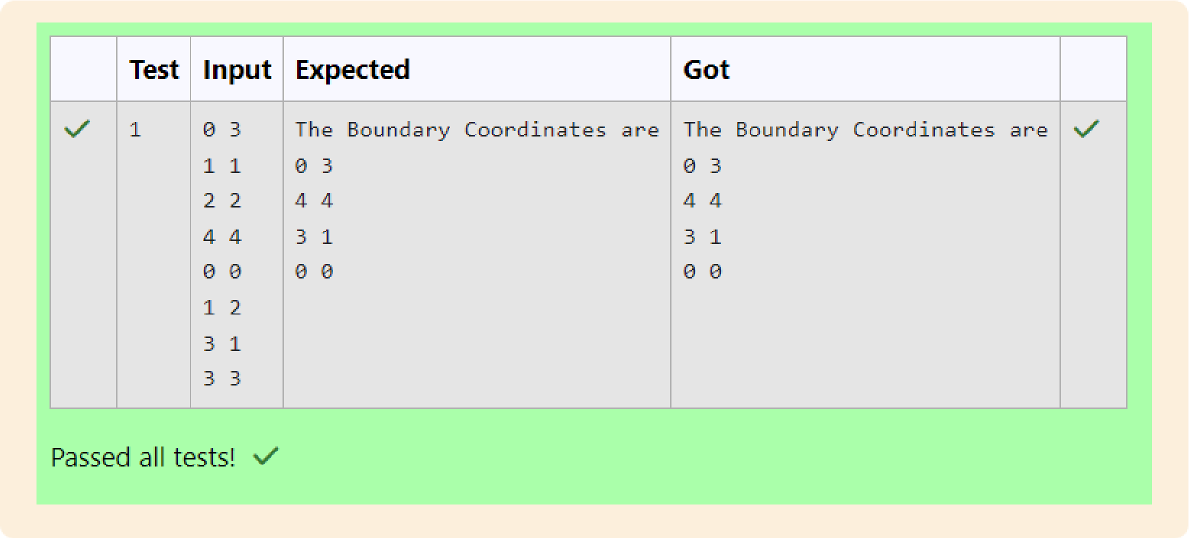


```
    int bd_index = 1;

    int prev = 0, curr = 1, next = 2;

    while (next < n + 1)
    {
        if (orientation(p[prev], p[curr], p[next]) == -1 || orientatio
        {
            // printf("{%d %d %d} Point %d - [%d %d] accepted %d\n", p
            border[bd_index] = curr;
            bd_index++;
            prev = curr;
            curr = next;
            next = next + 1;
            continue;
        }
        else if (orientation(p[prev], p[curr], p[next]) == 1) // i.e.
        {
```



```
            // printf("{%d %d %d} Point %d - [%d %d] rejected\n", prev
            curr = prev;
            bd_index--;
            prev = border[bd_index - 1];
        }
        // printf("%d %d %d <%d>\n", prev, curr, next, orientation(p[p
    }

    printf("The Boundary Coordinates are\n");
    for (int i = bd_index-1; i > 0; i--)
    {
        printf("%d %d\n", p[border[i]].x, p[border[i]].y);
    }
    printf("%d %d\n", p[border[0]].x, p[border[0]].y);
    return 0;
}
```

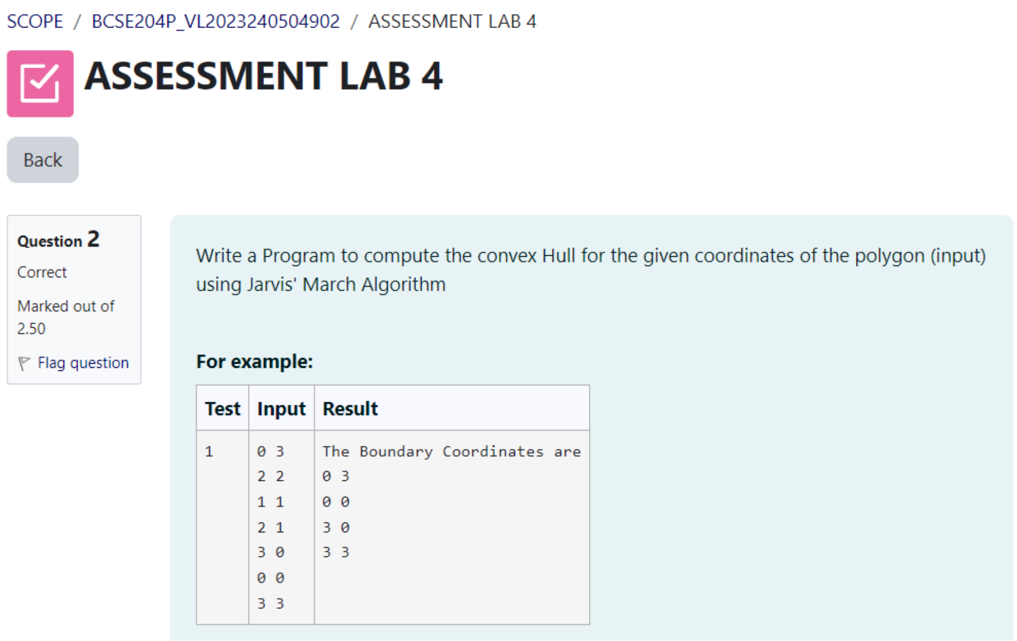
Output:



	Test	Input	Expected	Got	
✓	1	0 3 1 1 2 2 4 4 0 0 1 2 3 1 3 3	The Boundary Coordinates are 0 3 4 4 3 1 0 0	The Boundary Coordinates are 0 3 4 4 3 1 0 0	✓

Passed all tests! ✓

Question 2:



SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 4

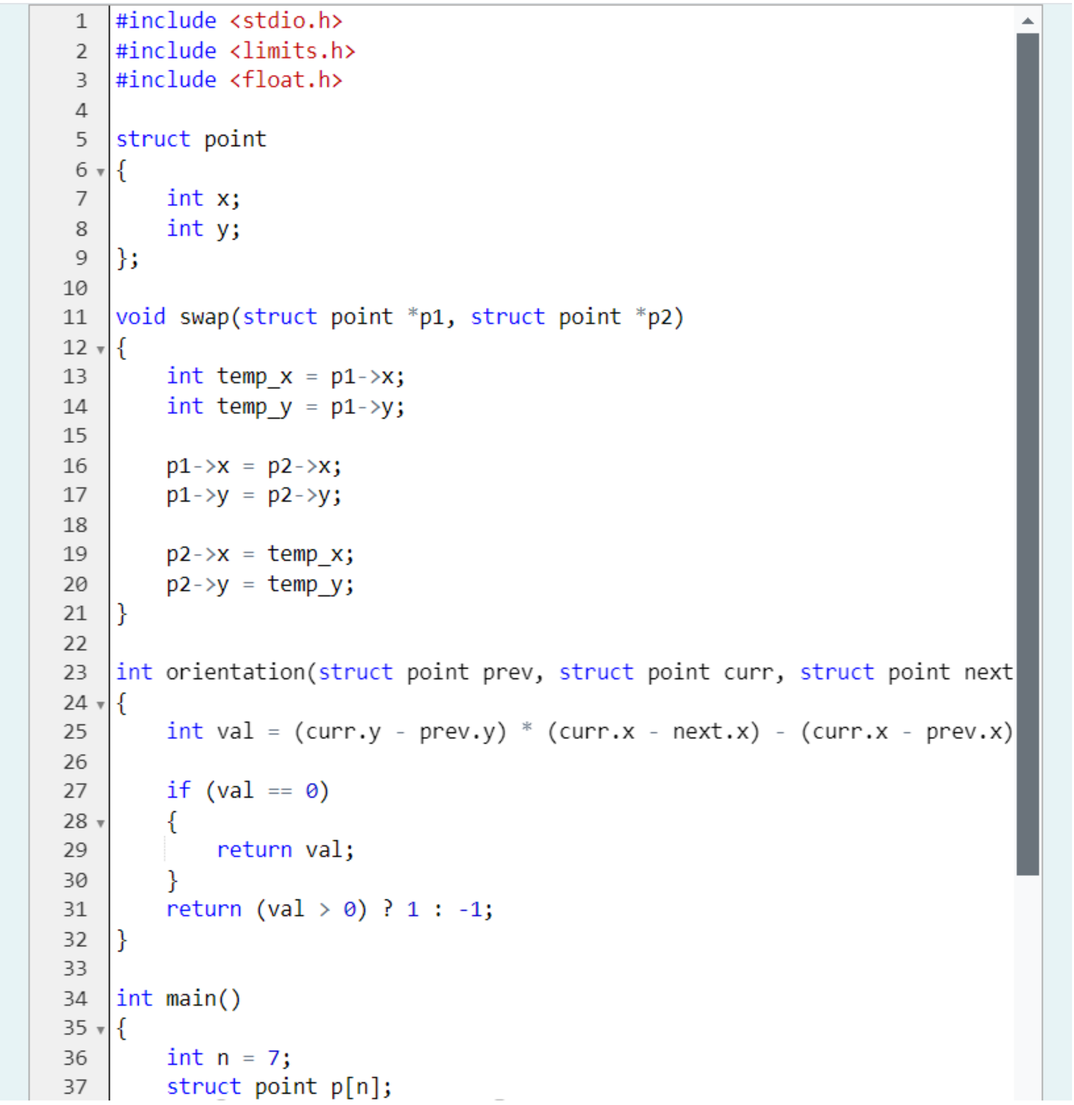
ASSESSMENT LAB 4

Write a Program to compute the convex Hull for the given coordinates of the polygon (input) using Jarvis' March Algorithm

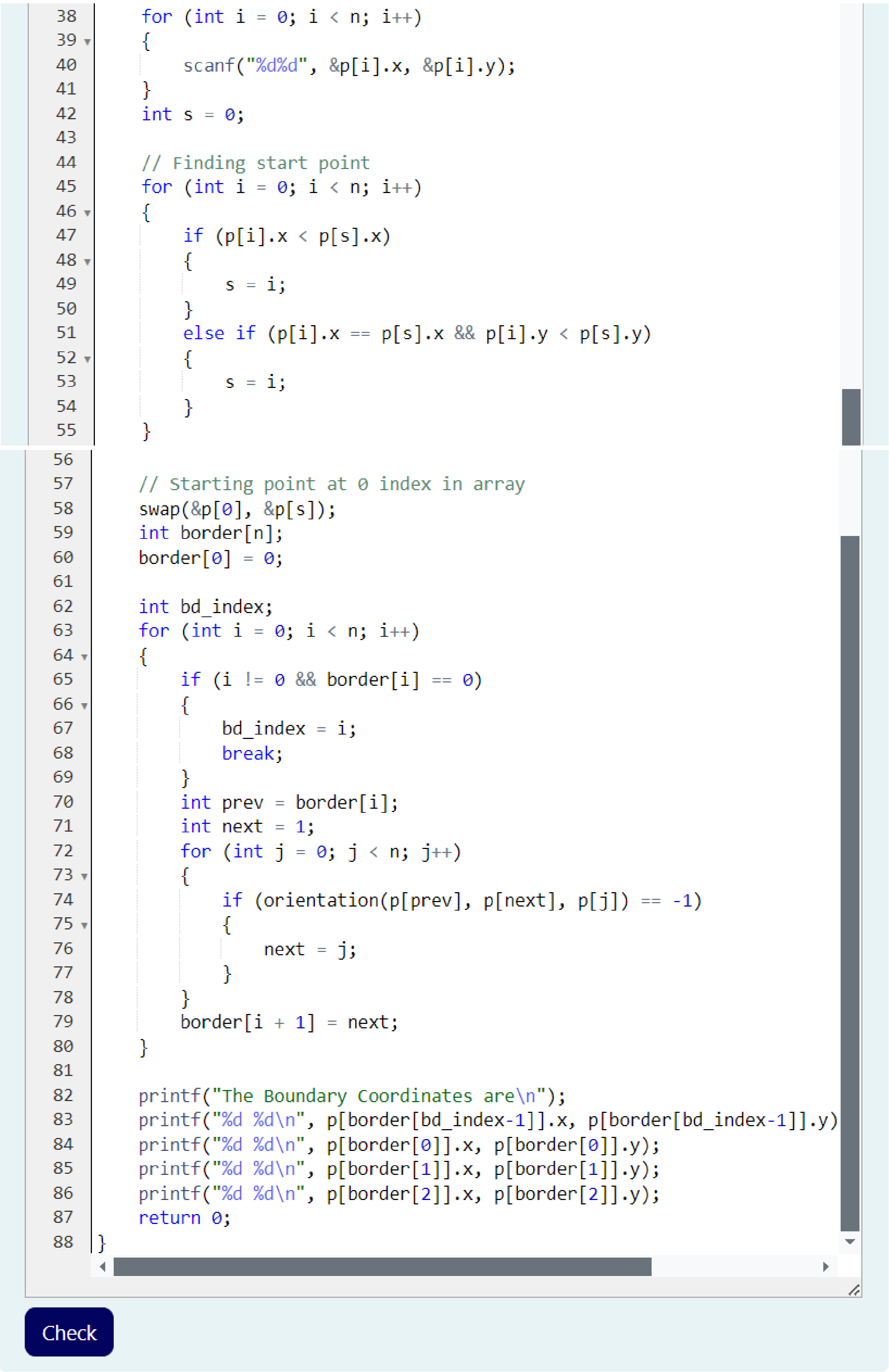
For example:

Test	Input	Result

1	0 3
	2 2
	1 1
	2 1
	3 0
	0 0
	3 3
		The Boundary Coordinates are
		0 3
		0 0
		3 0
		3 3

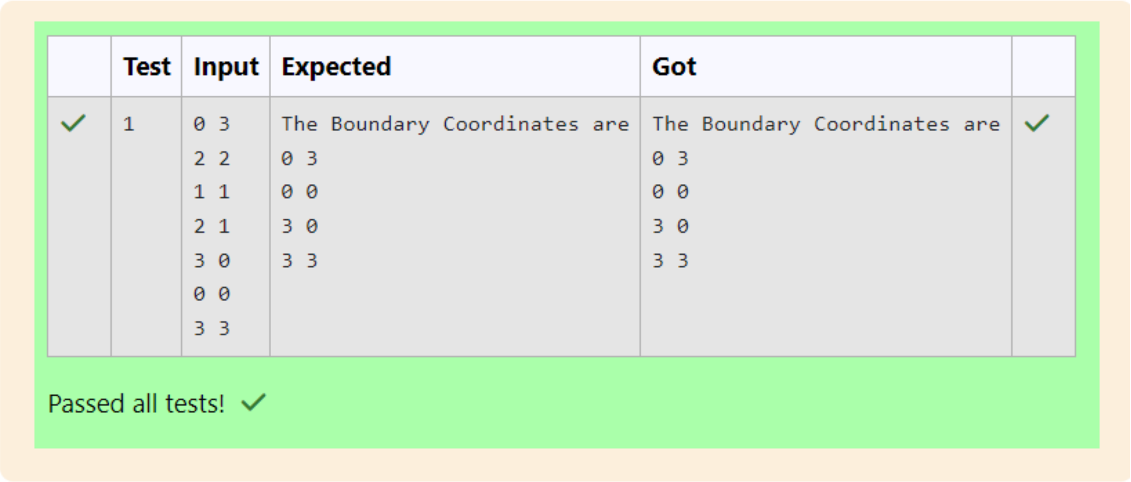


```
#include <stdio.h>
#include <limits.h>
#include <float.h>

struct point
{
    int x;
    int y;
};

void swap(struct point *p1, struct point *p2)
{
    int temp_x = p1->x;
    int temp_y = p1->y;

    p1->x = p2->x;
    p1->y = p2->y;

    p2->x = temp_x;
    p2->y = temp_y;
}

int orientation(struct point prev, struct point curr, struct point next
{
    int val = (curr.y - prev.y) * (curr.x - next.x) - (curr.x - prev.x)

    if (val == 0)
    {
        return val;
    }
    return (val > 0) ? 1 : -1;
}

int main()
{
    int n = 7;
    struct point p[n];
```



```c
    for (int i = 0; i < n; i++)
    {
        scanf("%d%d", &p[i].x, &p[i].y);
    }
    int s = 0;

    // Finding start point
    for (int i = 0; i < n; i++)
    {
        if (p[i].x < p[s].x)
        {
            s = i;
        }
        else if (p[i].x == p[s].x && p[i].y < p[s].y)
        {
            s = i;
        }
    }

    // Starting point at 0 index in array
    swap(&p[0], &p[s]);
    int border[n];
    border[0] = 0;

    int bd_index;
    for (int i = 0; i < n; i++)
    {
        if (i != 0 && border[i] == 0)
        {
            bd_index = i;
            break;
        }
        int prev = border[i];
        int next = 1;
        for (int j = 0; j < n; j++)
        {
            if (orientation(p[prev], p[next], p[j]) == -1)
            {
                next = j;
            }
        }
        border[i + 1] = next;
    }

    printf("The Boundary Coordinates are\n");
    printf("%d %d\n", p[border[bd_index-1]].x, p[border[bd_index-1]].y)
    printf("%d %d\n", p[border[0]].x, p[border[0]].y);
    printf("%d %d\n", p[border[1]].x, p[border[1]].y);
    printf("%d %d\n", p[border[2]].x, p[border[2]].y);
    return 0;
}
```

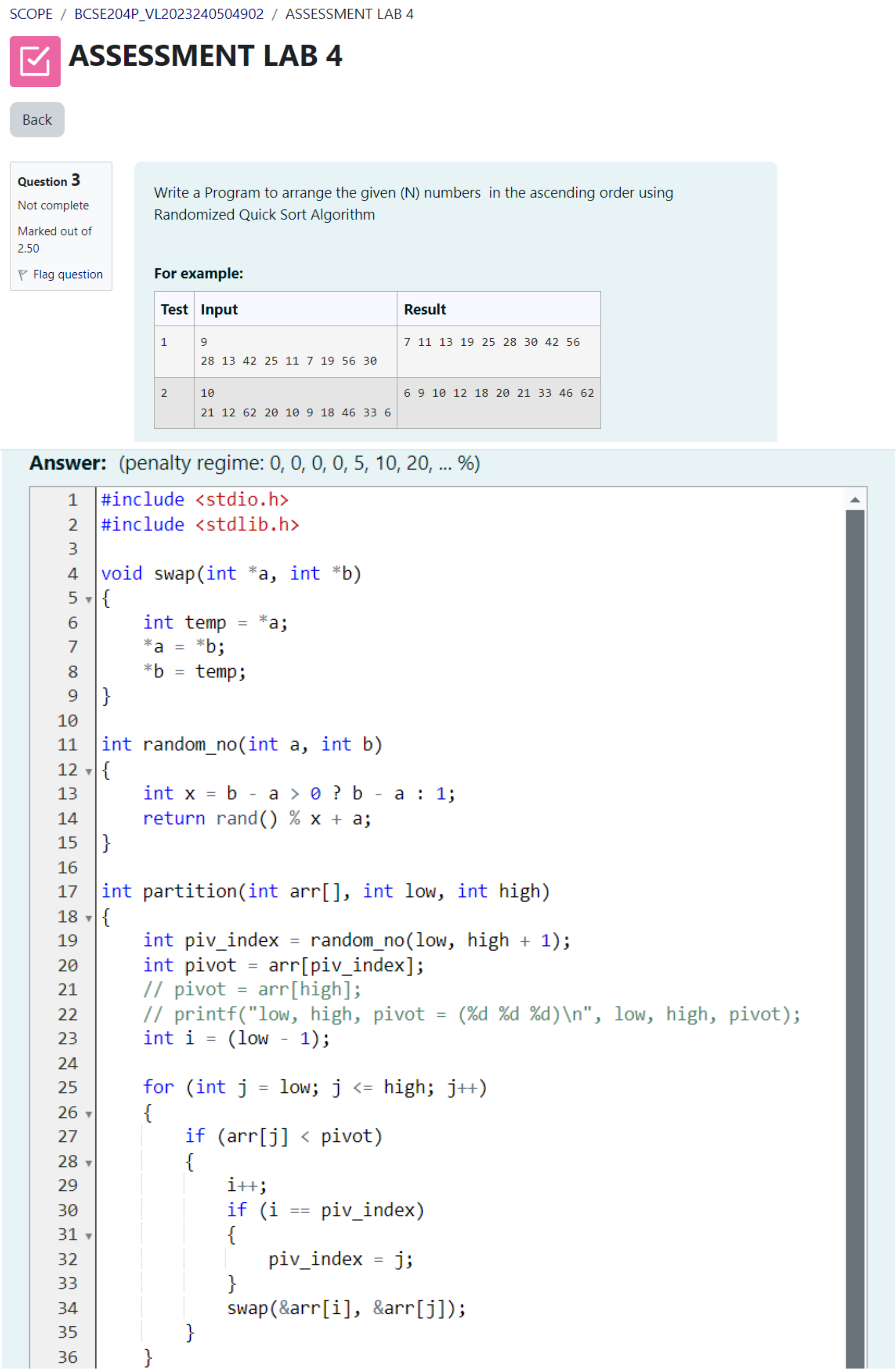
Output:



	Test	Input	Expected	Got	
✓	1	0 3 2 2 1 1 2 1 3 0 0 0 3 3	The Boundary Coordinates are 0 3 0 0 3 0 3 3	The Boundary Coordinates are 0 3 0 0 3 0 3 3	✓

Passed all tests! ✓

Question 3:



SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 4

ASSESSMENT LAB 4

Back

Question 3
Not complete
Marked out of 2.50
Flag question

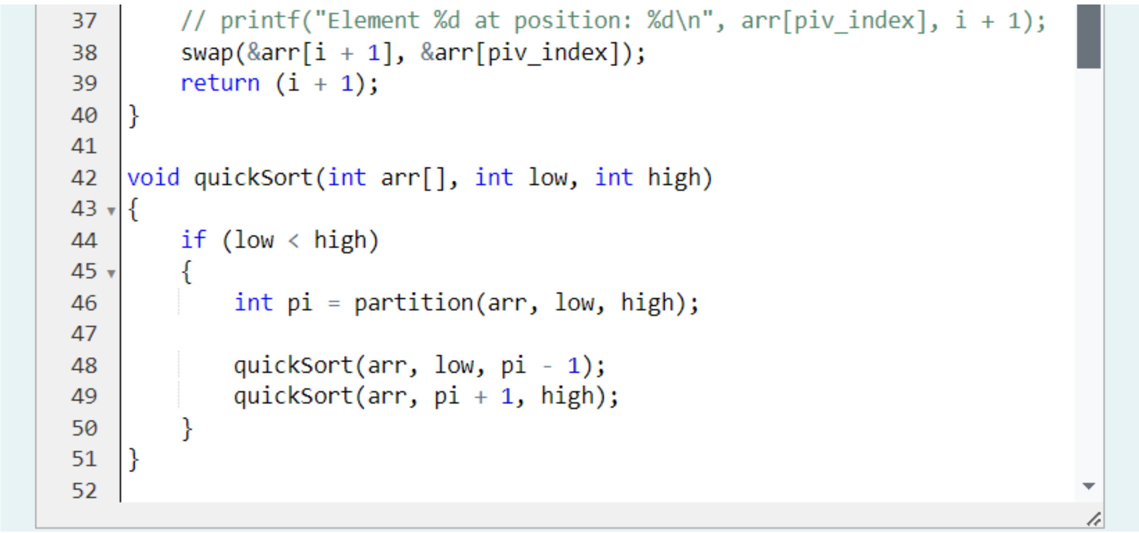
Write a Program to arrange the given (N) numbers in the ascending order using Randomized Quick Sort Algorithm

For example:

Test	Input	Result
1	9 28 13 42 25 11 7 19 56 30	7 11 13 19 25 28 30 42 56
2	10 21 12 62 20 10 9 18 46 33 6	6 9 10 12 18 20 21 33 46 62

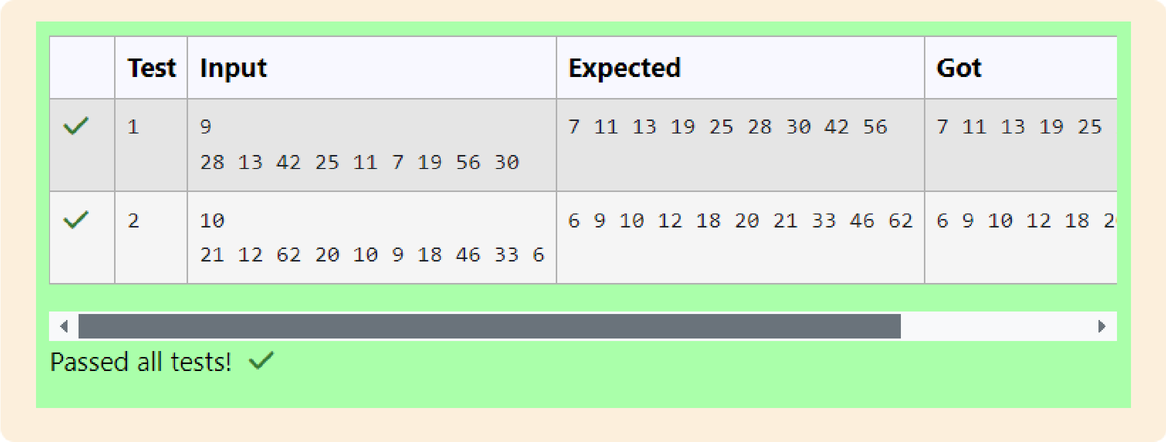
Answer: (penalty regime: 0, 0, 0, 0, 5, 10, 20, ... %)

```c
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int random_no(int a, int b)
{
    int x = b - a > 0 ? b - a : 1;
    return rand() % x + a;
}

int partition(int arr[], int low, int high)
{
    int piv_index = random_no(low, high + 1);
    int pivot = arr[piv_index];
    // pivot = arr[high];
    // printf("low, high, pivot = (%d %d %d)\n", low, high, pivot);
    int i = (low - 1);

    for (int j = low; j <= high; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            if (i == piv_index)
            {
                piv_index = j;
            }
            swap(&arr[i], &arr[j]);
        }
    }
```



```c
    // printf("Element %d at position: %d\n", arr[piv_index], i + 1);
    swap(&arr[i + 1], &arr[piv_index]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

```

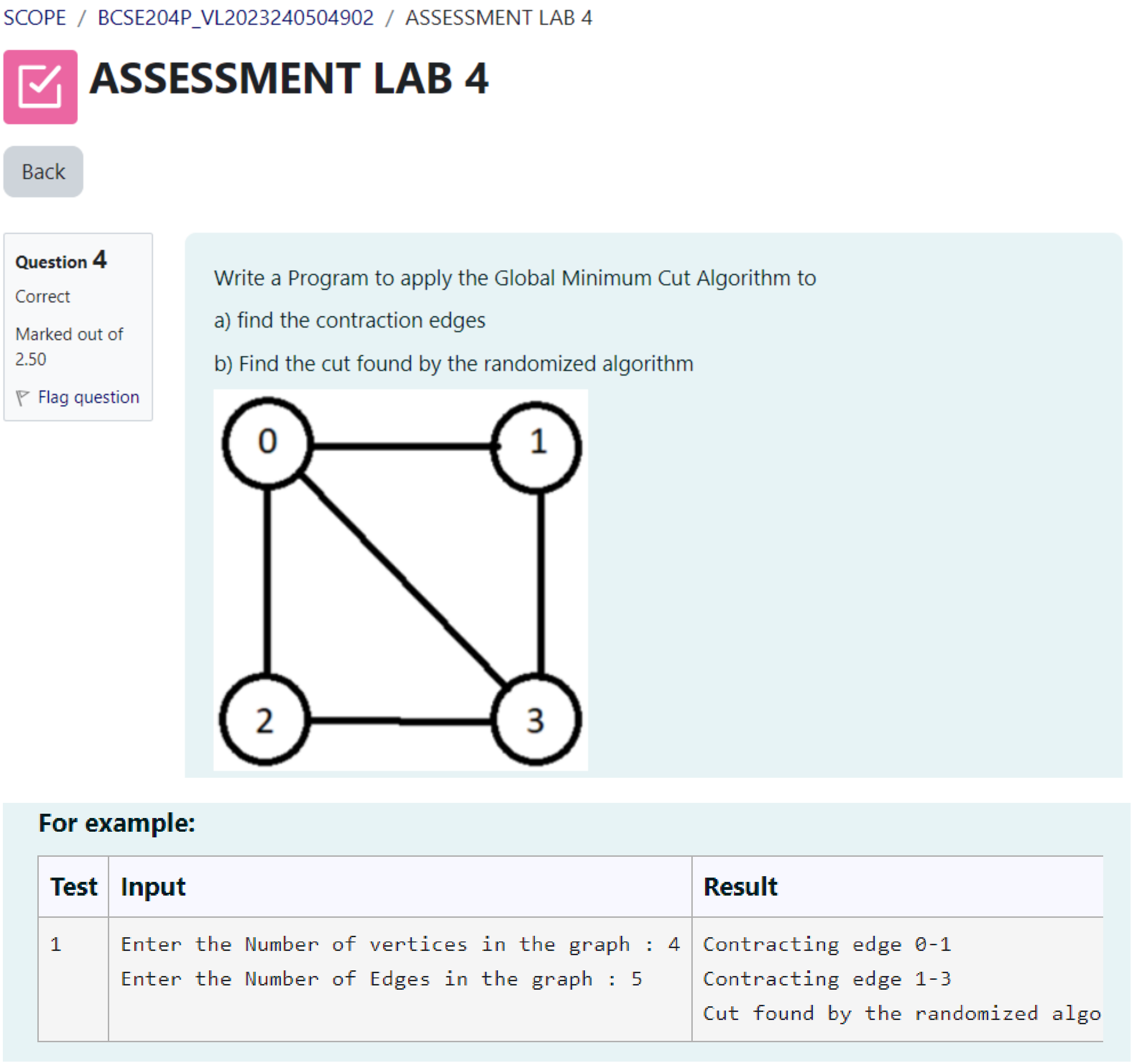
Output:



	Test	Input	Expected	Got
✓	1	9 28 13 42 25 11 7 19 56 30	7 11 13 19 25 28 30 42 56	7 11 13 19 25
✓	2	10 21 12 62 20 10 9 18 46 33 6	6 9 10 12 18 20 21 33 46 62	6 9 10 12 18 2

Passed all tests! ✓

Question 4:



SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 4

ASSESSMENT LAB 4

Back

Question 4

Correct

Marked out of 2.50

Flag question

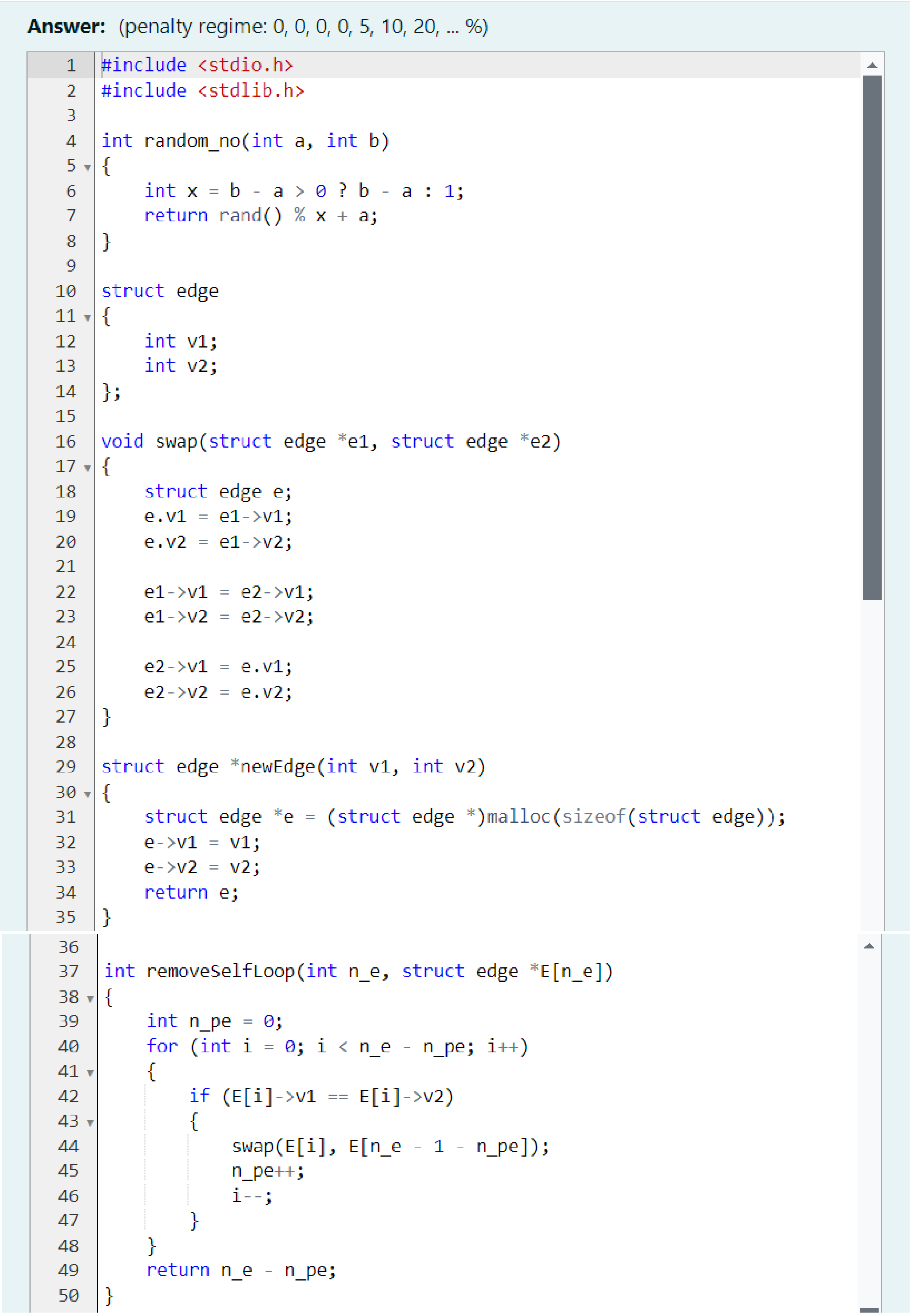
Write a Program to apply the Global Minimum Cut Algorithm to

a) find the contraction edges

b) Find the cut found by the randomized algorithm

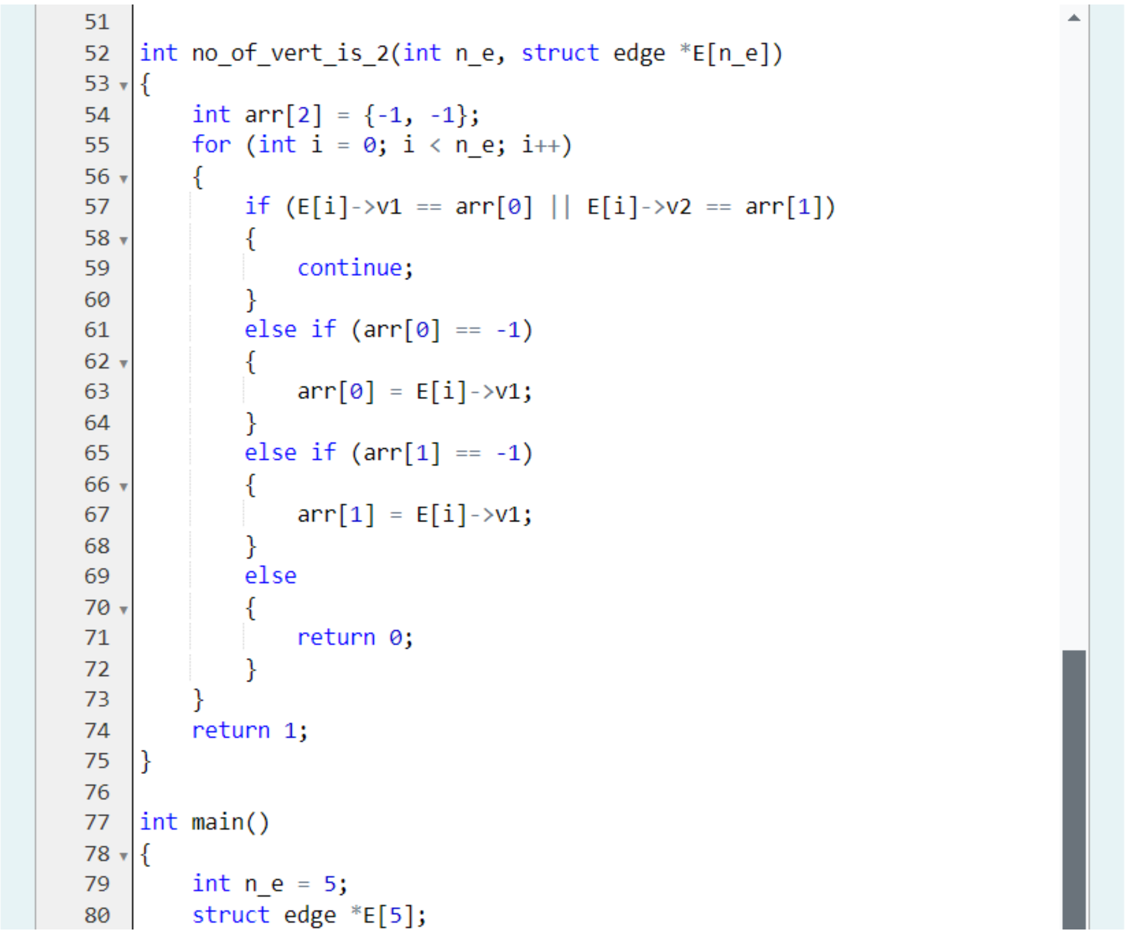
For example:

Test	Input	Result
1	Enter the Number of vertices in the graph : 4 Enter the Number of Edges in the graph : 5	Contracting edge 0-1 Contracting edge 1-3 Cut found by the randomized algo

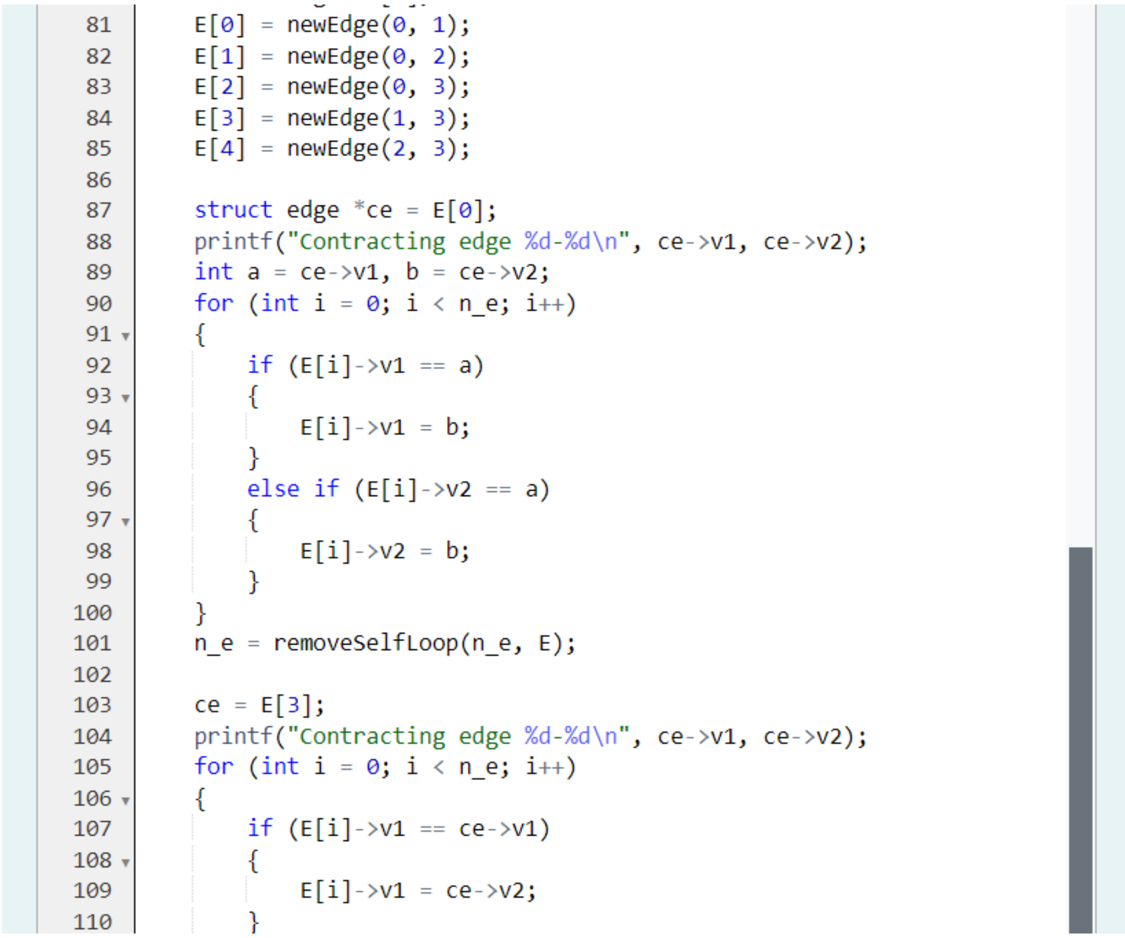


Answer: (penalty regime: 0, 0, 0, 0, 5, 10, 20, ... %)

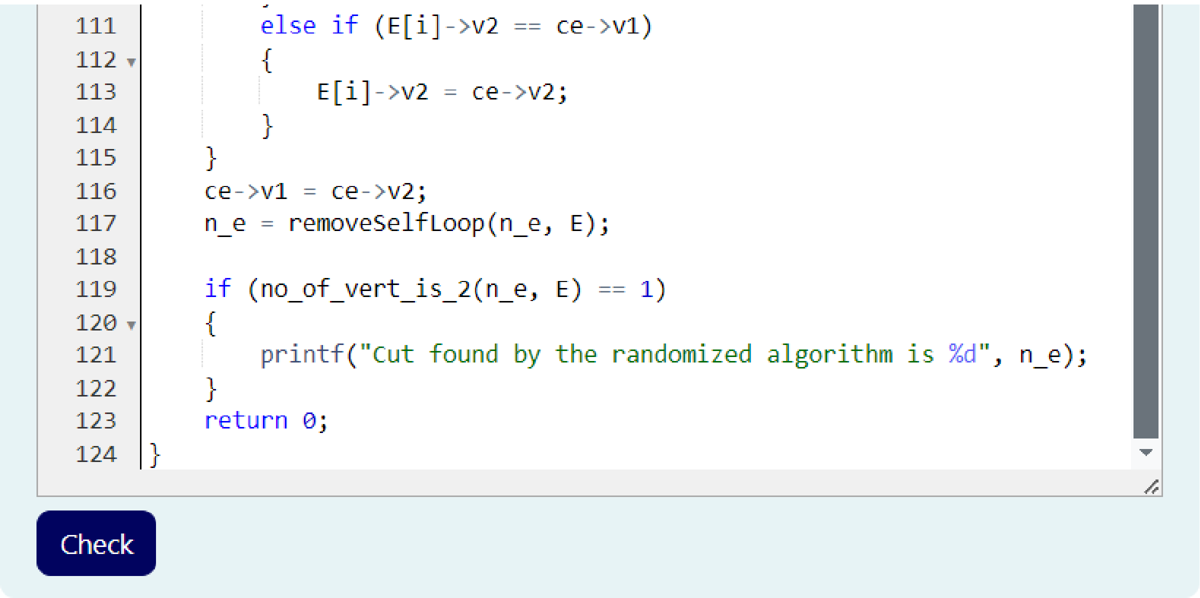
```c
#include <stdio.h>
#include <stdlib.h>

int random_no(int a, int b)
{
    int x = b - a > 0 ? b - a : 1;
    return rand() % x + a;
}

struct edge
{
    int v1;
    int v2;
};

void swap(struct edge *e1, struct edge *e2)
{
    struct edge e;
    e.v1 = e1->v1;
    e.v2 = e1->v2;

    e1->v1 = e2->v1;
    e1->v2 = e2->v2;

    e2->v1 = e.v1;
    e2->v2 = e.v2;
}

struct edge *newEdge(int v1, int v2)
{
    struct edge *e = (struct edge *)malloc(sizeof(struct edge));
    e->v1 = v1;
    e->v2 = v2;
    return e;
}

int removeSelfLoop(int n_e, struct edge *E[n_e])
{
    int n_pe = 0;
    for (int i = 0; i < n_e - n_pe; i++)
    {
        if (E[i]->v1 == E[i]->v2)
        {
            swap(E[i], E[n_e - 1 - n_pe]);
            n_pe++;
            i--;
        }
    }
    return n_e - n_pe;
}
```



```c

int no_of_vert_is_2(int n_e, struct edge *E[n_e])
{
    int arr[2] = {-1, -1};
    for (int i = 0; i < n_e; i++)
    {
        if (E[i]->v1 == arr[0] || E[i]->v2 == arr[1])
        {
            continue;
        }
        else if (arr[0] == -1)
        {
            arr[0] = E[i]->v1;
        }
        else if (arr[1] == -1)
        {
            arr[1] = E[i]->v1;
        }
        else
        {
            return 0;
        }
    }
    return 1;
}

int main()
{
    int n_e = 5;
    struct edge *E[5];
```

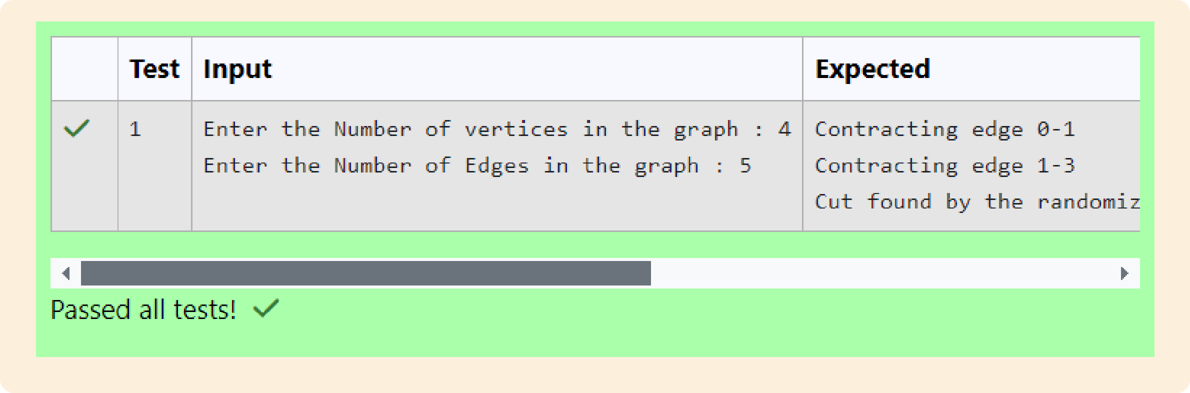


```c
    E[0] = newEdge(0, 1);
    E[1] = newEdge(0, 2);
    E[2] = newEdge(0, 3);
    E[3] = newEdge(1, 3);
    E[4] = newEdge(2, 3);

    struct edge *ce = E[0];
    printf("Contracting edge %d-%d\n", ce->v1, ce->v2);
    int a = ce->v1, b = ce->v2;
    for (int i = 0; i < n_e; i++)
    {
        if (E[i]->v1 == a)
        {
            E[i]->v1 = b;
        }
        else if (E[i]->v2 == a)
        {
            E[i]->v2 = b;
        }
    }
    n_e = removeSelfLoop(n_e, E);

    ce = E[3];
    printf("Contracting edge %d-%d\n", ce->v1, ce->v2);
    for (int i = 0; i < n_e; i++)
    {
        if (E[i]->v1 == ce->v1)
        {
            E[i]->v1 = ce->v2;
        }
```



```
        else if (E[i]->v2 == ce->v1)
        {
            E[i]->v2 = ce->v2;
        }
    }
    ce->v1 = ce->v2;
    n_e = removeSelfLoop(n_e, E);

    if (no_of_vert_is_2(n_e, E) == 1)
    {
        printf("Cut found by the randomized algorithm is %d", n_e);
    }
    return 0;
}
```

Output:



	Test	Input	Expected
✓	1	Enter the Number of vertices in the graph : 4 Enter the Number of Edges in the graph : 5	Contracting edge 0-1 Contracting edge 1-3 Cut found by the randomiz

Passed all tests! ✓