Foundation of Algorithms

Assignment

IT19145044

Ovitigala N.M.O

CS Y2S2

Contents

Question 1

You have been given 1D integer array of size N . In addition, you have been given an element M and you need to find and print the index of the last occurrence of this element M in the array if it exists in it, otherwise print -1.

Input:

The first line consists of 2 integers N and M denoting the size of the array and the element

to be searched for in the array respectively.

Second line contains N space separated integers denoting the elements of the array.

Output:

Print a single integer denoting the index of the last occurrence of integer M in the array if

it exists, otherwise print -1.

Sample Input	Sample Output
5 1	5
1 2 3 4 1	

Source Code

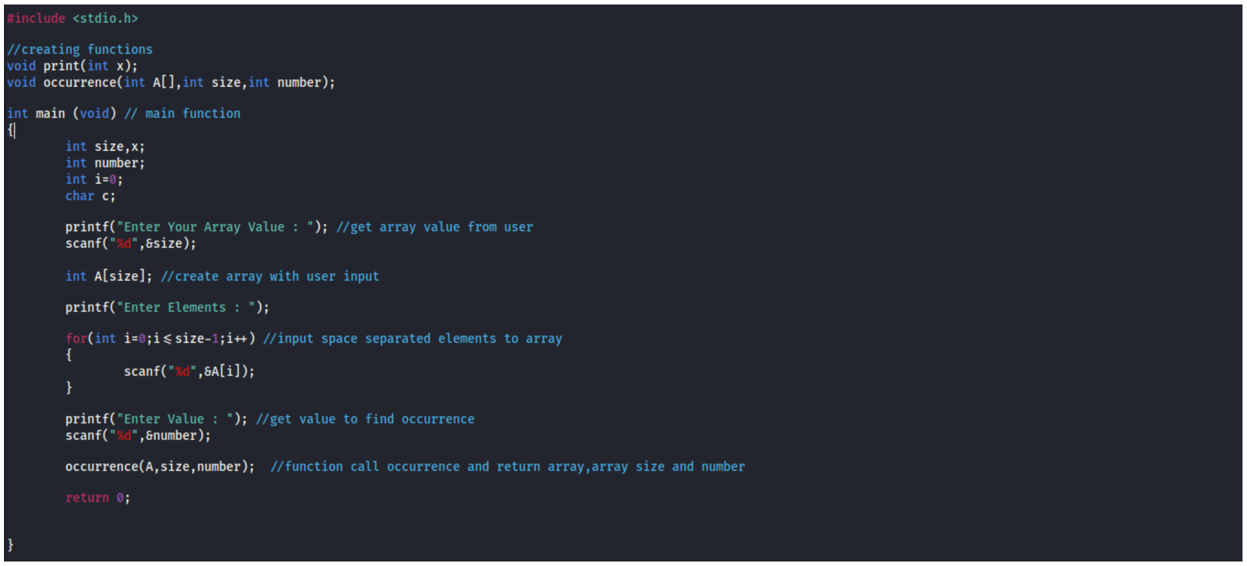
```
#include <stdio.h>

//creating functions
void print(int x);
void occurrence(int A[],int size,int number);

int main (void) // main function
{
        int size;
        int number;

        printf("Enter Your Array Value : "); //get array value from user
        scanf("%d",&size);

        int A[size]; //create array with user input size

        printf("Enter Elements : ");

        for(int i=0;i<=size-1;i++) //input space separated elements to array
        {
                scanf("%d",&A[i]);
        }

        printf("Enter Value : "); //get value to find occurrence
        scanf("%d",&number);
```

```
            occurrence(A,size,number);  //function call occurrence and return array,array size and
number
            return 0;
}

void print(int x) //print function
{
            //checking x value
            if(x==0)
            {
                        printf("-1\n"); //no occurrence, so print -1
            }
            else
            {
                        printf("Last Occurrence is : %d\n",x+1); // print last occurrence
            }
}

void occurrence(int A[],int size,int number) //occurrence function
{
            int x;

            for(int i=0;i<size;i++)
            {
                        if(number==A[i]) //if number value equal to A[i] value then i value add to x
variable
                        {
```

```
                    x=i;
                }

            }

            print(x); //function call print and return x value

        }
```

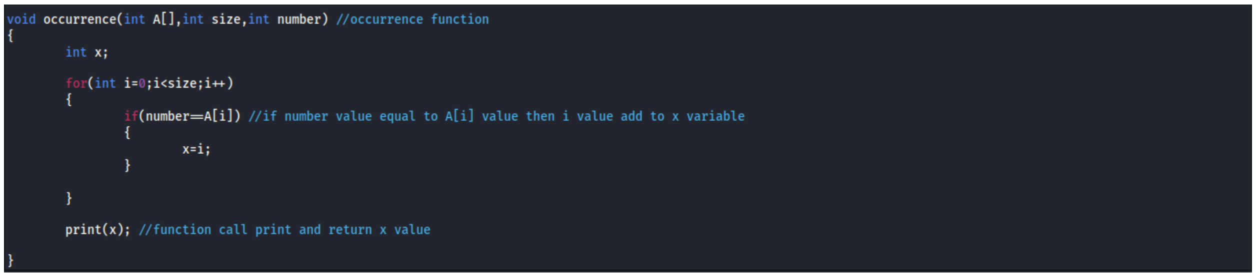
Screen Shots of the Source Code

main function



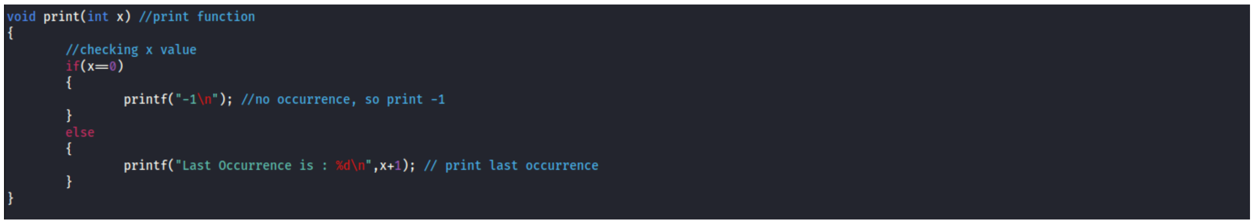
```c
#include <stdio.h>

//creating functions
void print(int x);
void occurrence(int A[],int size,int number);

int main (void) // main function
{
	int size,x;
	int number;
	int i=0;
	char c;

	printf("Enter Your Array Value : "); //get array value from user
	scanf("%d",&size);

	int A[size]; //create array with user input

	printf("Enter Elements : ");

	for(int i=0;i<=size-1;i++) //input space separated elements to array
	{
		scanf("%d",&A[i]);
	}

	printf("Enter Value : "); //get value to find occurrence
	scanf("%d",&number);

	occurrence(A,size,number);  //function call occurrence and return array,array size and number

	return 0;


}
```

occurrence funtion



```c
void occurrence(int A[],int size,int number) //occurrence function
{
	int x;

	for(int i=0;i<size;i++)
	{
		if(number==A[i]) //if number value equal to A[i] value then i value add to x variable
		{
			x=i;
		}

	}

	print(x); //function call print and return x value

}
```

print function



```c
void print(int x) //print function
{
	//checking x value
	if(x==0)
	{
		printf("-1\n"); //no occurrence, so print -1
	}
	else
	{
		printf("Last Occurrence is : %d\n",x+1); // print last occurrence
	}
}
```

Description

main function

- Getting array value from user
- Creating array using user input size
- Getting input space separated elements to array
- Getting value to find occurrence
- Function call occurrence with return array name, array size, and number

occurrence function

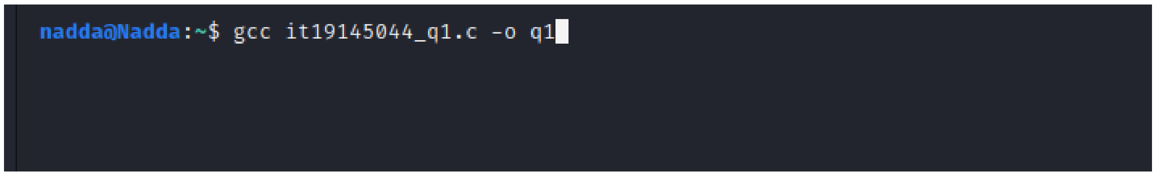
- Checking if the number equal to A[i] value then i value add to x variable
- Function call print using x value

print function

- Checking x value
- If no occurrence (x==0) print -1
- If it has a occurrence print it

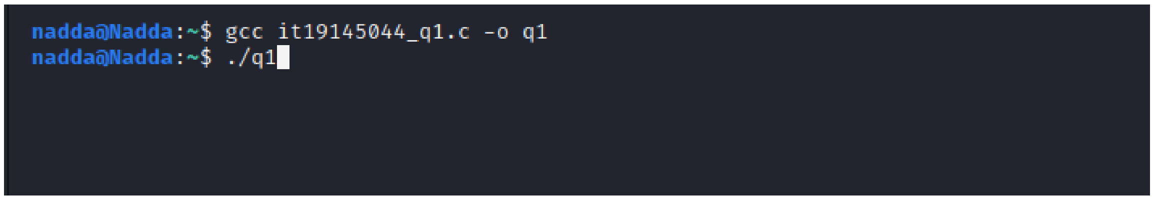
Sample Outputs

Compile source code : gcc sample.c -o sample



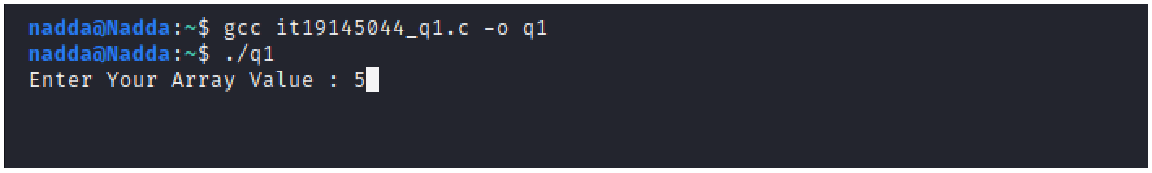
```
nadda@Nadda:~$ gcc it19145044_q1.c -o q1
```

Run compile file : ./sample



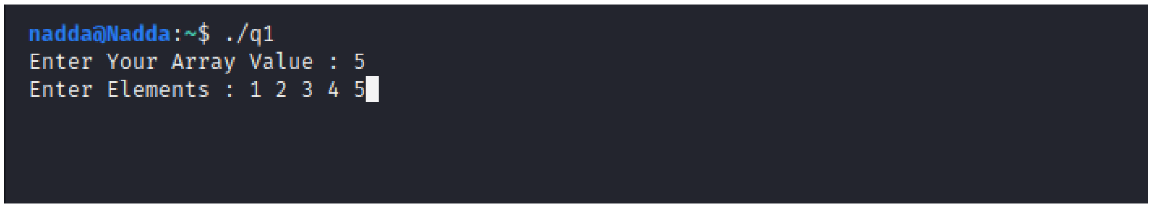
```
nadda@Nadda:~$ gcc it19145044_q1.c -o q1
nadda@Nadda:~$ ./q1
```

Input Array size : 5



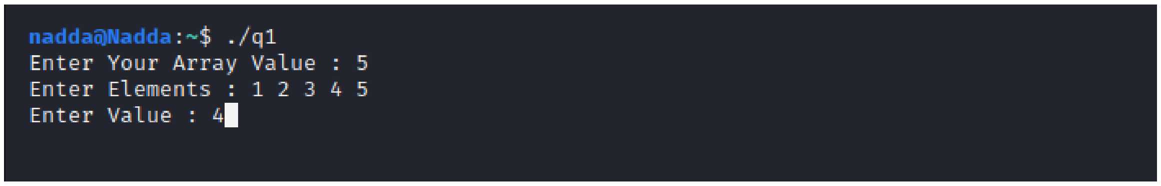
```
nadda@Nadda:~$ gcc it19145044_q1.c -o q1
nadda@Nadda:~$ ./q1
Enter Your Array Value : 5
```

Input elements : 1 2 3 4 5



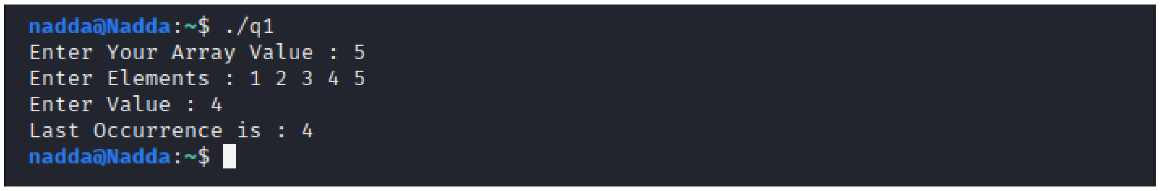
```
nadda@Nadda:~$ ./q1
Enter Your Array Value : 5
Enter Elements : 1 2 3 4 5
```

Input value to find occurrence : 4



```
nadda@Nadda:~$ ./q1
Enter Your Array Value : 5
Enter Elements : 1 2 3 4 5
Enter Value : 4
```

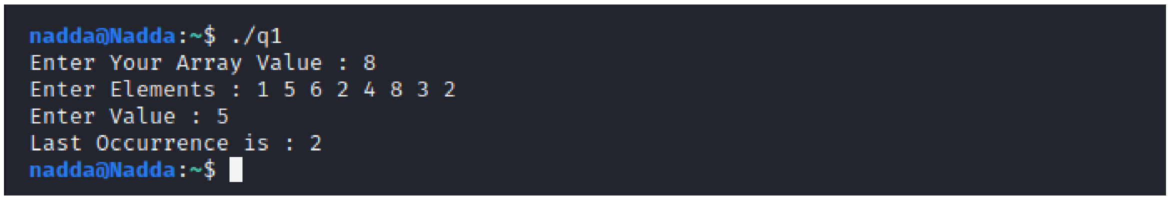
Output : 4



```
nadda@Nadda:~$ ./q1
Enter Your Array Value : 5
Enter Elements : 1 2 3 4 5
Enter Value : 4
Last Occurrence is : 4
nadda@Nadda:~$
```

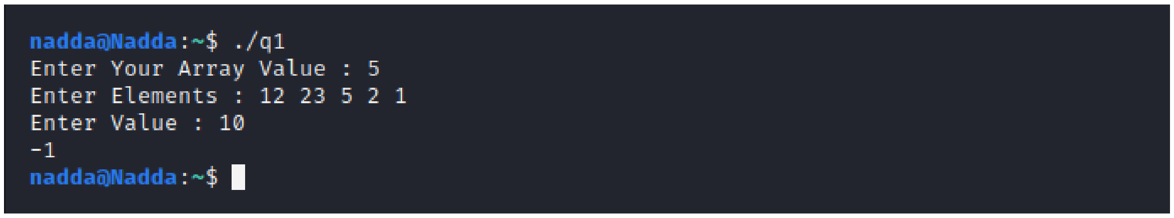
Examples

1. Array Size : 8
   Elements : 1 5 6 2 4 8 3 2
   Value : 5



```
nadda@Nadda:~$ ./q1
Enter Your Array Value : 8
Enter Elements : 1 5 6 2 4 8 3 2
Enter Value : 5
Last Occurrence is : 2
nadda@Nadda:~$
```

2. Array Size : 5
   Elements : 12 23 5 2 1
   Value : 10



```
nadda@Nadda:~$ ./q1
Enter Your Array Value : 5
Enter Elements : 12 23 5 2 1
Enter Value : 10
-1
nadda@Nadda:~$
```

Question 2

Given an array A on size N, you need to find the number of ordered pairs (i,j) such that

i<j and A[i] > A[j].

Input:

First line contains one integer N, the size of array.

Second line contains N space separated integers denoting the elements of the array A.

Output:

First line prints the number of ordered pairs (i,j) such that i<j and A[i] > A[j].

Second line prints those ordered pairs.

Hint: Use Divide & Conquer method

Sample inputs & outputs are as follows.

Sample Input	**Sample Output**
5	3
1 4 3 2 5	(4,3) (4,2) (3,2)

Source Code

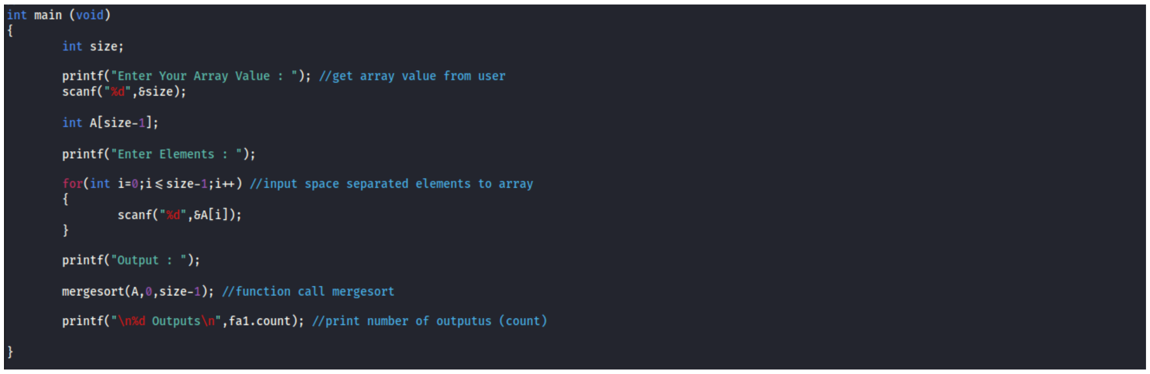
```
#include <stdio.h>

#include <stdlib.h>



struct fa  // create structer call "fa"

{

        int count;

}fa1;


// create functions

void mergesort(int A[],int p,int r);

void merge(int A[],int p, int q, int r);

void print(int x, int y);


//main function

int main (void)

{

        int size;


        printf("Enter Your Array Value : "); //get array value from user

        scanf("%d",&size);


        int A[size-1];


        printf("Enter Elements : ");


        for(int i=0;i<=size-1;i++) //input space separated elements to array
        {
                scanf("%d",&A[i]);
        }

        printf("Output : ");

        mergesort(A,0,size-1); //function call mergesort

        printf("\n%d Outputs\n",fa1.count); //print number of outputus (count)

}

void print(int x, int y) //print function
{
        fa1.count++; // count number of outputs
        printf("(%d,%d)",x,y); //print pairs
}

void merge(int A[],int p, int q, int r)  //merge function
{
        int i;
        int j;
        int k;
        int new1 = (q-p) + 1;
        int new2 = (r-q);

        int left[new1], right[new2]; //create temporary arrays
```

```
for (i=0;i<new1;i++) //Copy data to temporary arrays left[] and right[]
{
        left[i]=A[p+i];
}
for (j=0;j<new2;j++)
{
        right[j] = A[q+j+1];
}

i=0;
j=0;
k=p;

while (i<new1 && j<new2) //check left side array value and right side arra value
{
        if (left[i]>right[j]) //check left side array value grater than right side array value
        {
                A[k]=left[i];
                print(left[i],right[j]); // print function call
                i++;

        }
        else
        {
                A[k] = right[j];
                j++;
        }

        k++;
        }

        while (i<new1) // copy remaining element of left side
        {
                A[k]=left[i];
                i++;
                k++;
        }

        while (j<new2) // copy remaining element of right side
        {
                A[k]=right[j];
                j++;
                k++;
        }
}

void mergesort(int A[],int p,int r) //mergesort function
{
        if(p<r) // check array last value grater than first value
        {
                int q= p + (r-p) /2; //find partition value

                mergesort(A,p,q); //apply mergesort function to left side array until can't devide
                mergesort(A,q+1,r); //apply mergesort function to right side array until can't devide
```

```
                    merge(A,p,q,r); // merge left side array and right side array
          }


}
```

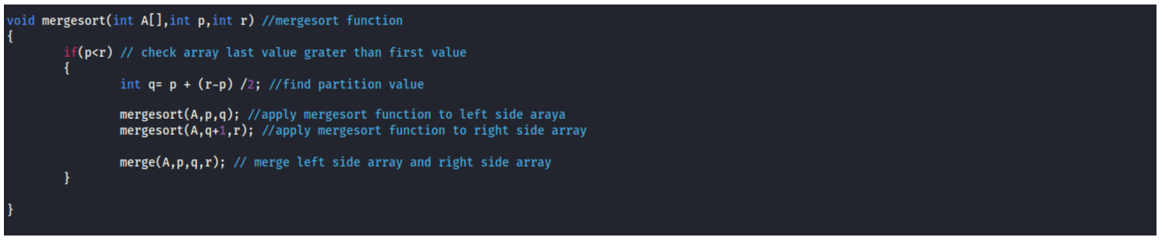
Screen Shots of the Source Code

main function



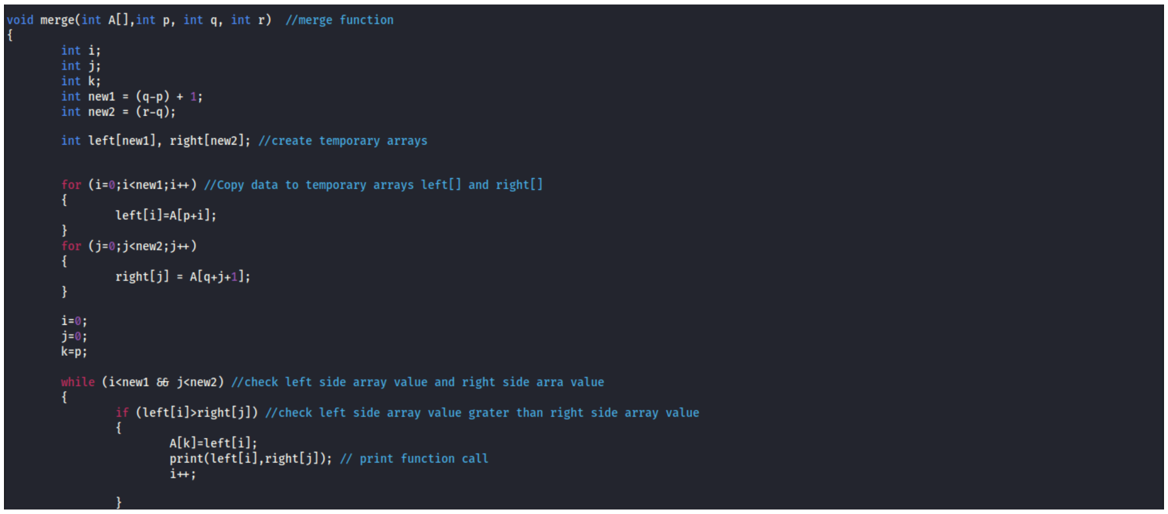
```c
int main (void)
{
        int size;

        printf("Enter Your Array Value : "); //get array value from user
        scanf("%d",&size);

        int A[size-1];

        printf("Enter Elements : ");

        for(int i=0;i<=size-1;i++) //input space separated elements to array
        {
                scanf("%d",&A[i]);
        }

        printf("Output : ");

        mergesort(A,0,size-1); //function call mergesort

        printf("\n%d Outputs\n",fa1.count); //print number of outputus (count)

}
```

mergesort function

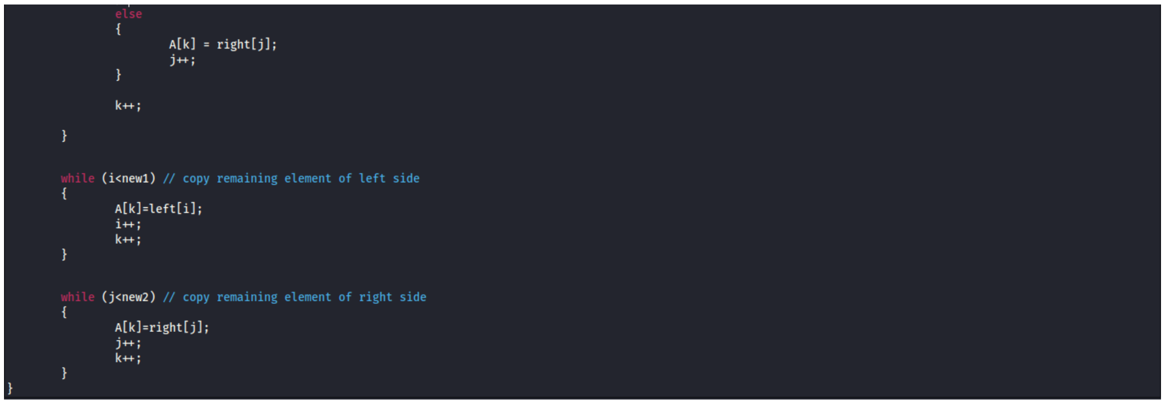


```c
void mergesort(int A[],int p,int r) //mergesort function
{
        if(p<r) // check array last value grater than first value
        {
                int q= p + (r-p) /2; //find partition value

                mergesort(A,p,q); //apply mergesort function to left side araya
                mergesort(A,q+1,r); //apply mergesort function to right side array

                merge(A,p,q,r); // merge left side array and right side array
        }

}
```

merge function

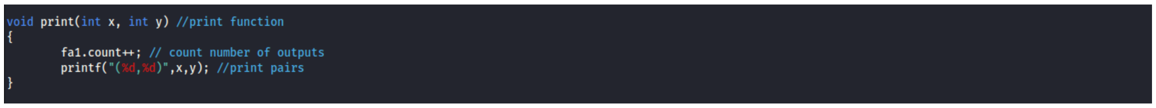


```c
void merge(int A[],int p, int q, int r)  //merge function
{
        int i;
        int j;
        int k;
        int new1 = (q-p) + 1;
        int new2 = (r-q);

        int left[new1], right[new2]; //create temporary arrays


        for (i=0;i<new1;i++) //Copy data to temporary arrays left[] and right[]
        {
                left[i]=A[p+i];
        }
        for (j=0;j<new2;j++)
        {
                right[j] = A[q+j+1];
        }

        i=0;
        j=0;
        k=p;

        while (i<new1 && j<new2) //check left side array value and right side arra value
        {
                if (left[i]>right[j]) //check left side array value grater than right side array value
                {
                        A[k]=left[i];
                        print(left[i],right[j]); // print function call
                        i++;

                }
```



```
                else
                {
                        A[k] = right[j];
                        j++;
                }

                k++;

        }


        while (i<new1) // copy remaining element of left side
        {
                A[k]=left[i];
                i++;
                k++;
        }


        while (j<new2) // copy remaining element of right side
        {
                A[k]=right[j];
                j++;
                k++;
        }
}
```

print function



```
void print(int x, int y) //print function
{
        fa1.count++; // count number of outputs
        printf("(%d,%d)",x,y); //print pairs
}
```

Description

main function

- Getting array value from user
- Create Array using user input size
- Getting input space separated elements to array
- Function call mergesort with array name,number of start point and array size
- Print number of outputus

mergesort function

- check array last value grater than first value
- find partition value
- apply mergesort function to left side array until can’t devide
- apply mergesort function to right side array until can’t devide
- function call merge

merge function

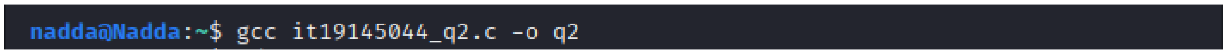
- Create temporary arrays
- Copying data to temporary arrays left[] and right[]
- Check left side array value grater than right side array value
- Function call print
- Copying remaining element of left side and right side

print function

- counting number of outputs
- print pairs

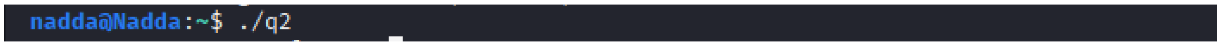
Sample Outputs

Compile source code : gcc sample.c -o sample



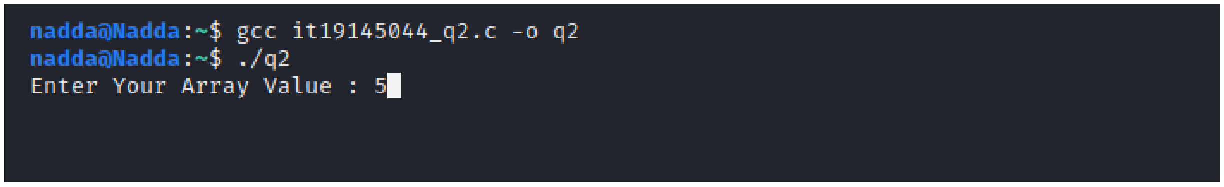
```
nadda@Nadda:~$ gcc it19145044_q2.c -o q2
```

Run compile file : ./sample



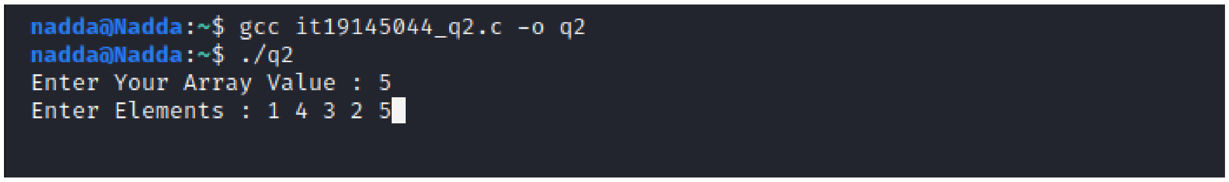
```
nadda@Nadda:~$ ./q2
```

Input Array Size : 5



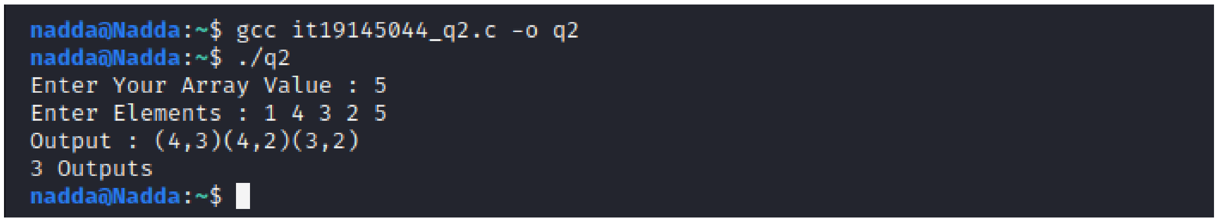
```
nadda@Nadda:~$ gcc it19145044_q2.c -o q2
nadda@Nadda:~$ ./q2
Enter Your Array Value : 5
```

Input Array Values : 1 4 3 2 5



```
nadda@Nadda:~$ gcc it19145044_q2.c -o q2
nadda@Nadda:~$ ./q2
Enter Your Array Value : 5
Enter Elements : 1 4 3 2 5
```

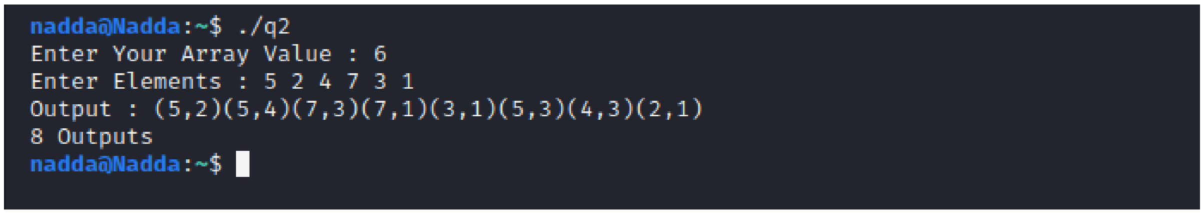
Output : (4,3)(4,2)(3,2)



```
nadda@Nadda:~$ gcc it19145044_q2.c -o q2
nadda@Nadda:~$ ./q2
Enter Your Array Value : 5
Enter Elements : 1 4 3 2 5
Output : (4,3)(4,2)(3,2)
3 Outputs
nadda@Nadda:~$
```

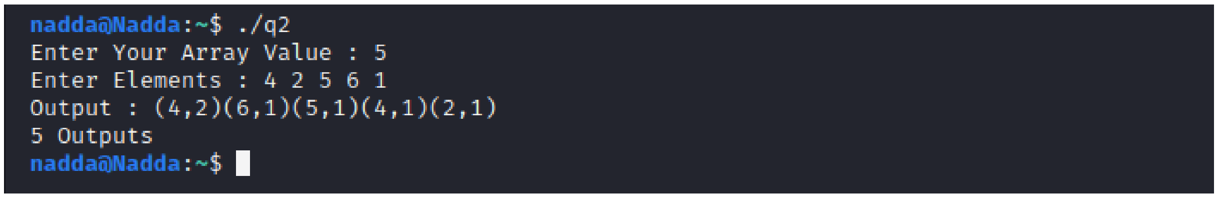
Examples

1. Array Size : 6
   Elements : 5 2 4 7 3 1



```
nadda@Nadda:~$ ./q2
Enter Your Array Value : 6
Enter Elements : 5 2 4 7 3 1
Output : (5,2)(5,4)(7,3)(7,1)(3,1)(5,3)(4,3)(2,1)
8 Outputs
nadda@Nadda:~$
```

2. Array Size : 5
   Elements : 4 2 5 6 1



```
nadda@Nadda:~$ ./q2
Enter Your Array Value : 5
Enter Elements : 4 2 5 6 1
Output : (4,2)(6,1)(5,1)(4,1)(2,1)
5 Outputs
nadda@Nadda:~$
```