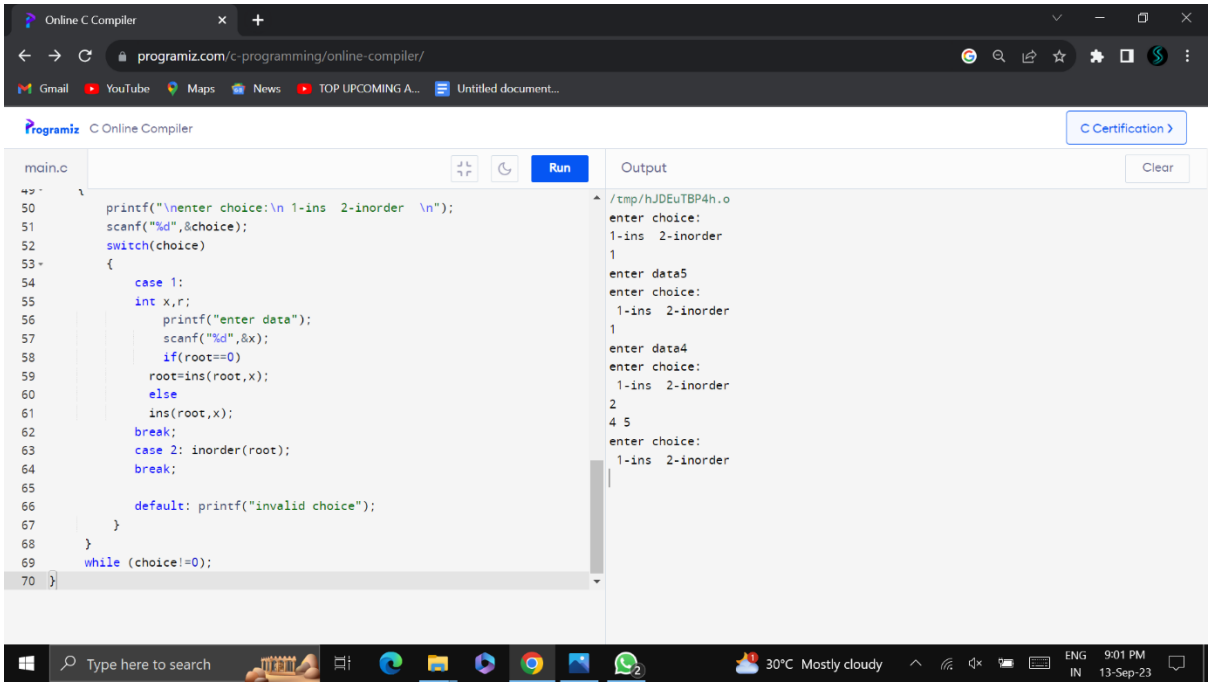
# DATA STRUCTURES

1. Write a c program to implement binary search tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node {
  int data;
  struct node *r;
  struct node *l;
};
void inorder(struct node* root) {
    if (root != 0) {
        inorder(root->l);
        printf("%d ", root->data);
        inorder(root->r);
    }
}
struct node* create(int x)
{
  struct node *nn;
  nn =(struct node*) malloc(sizeof(struct node));
  nn->data = x;
  nn->l = 0;
  nn->r= 0;

  return nn;
}
```

```
struct node* ins(struct node *root,int x )
{
if(root==0)
{
	return create(x);
}
	if(x<root->data)
	{
		root->l=ins(root->l,x);
	}
	else
	{
		root->r=ins(root->r,x);
	}
	return root;
}

int main()
{
	struct node *root=0;

	int choice;
	do
	{
		printf("\nenter choice:\n 1-ins  2-inorder  \n");
		scanf("%d",&choice);
		switch(choice)
```

```
                {
                        case 1:
                        int x,r;
        printf("enter data");
        scanf("%d",&x);
        if(root==0)
                         root=ins(root,x);
                         else
                         ins(root,x);
                        break;
                        case 2: inorder(root);
                        break;

                        default: printf("invalid choice");
                }
        }
        while (choice!=0);
}
```



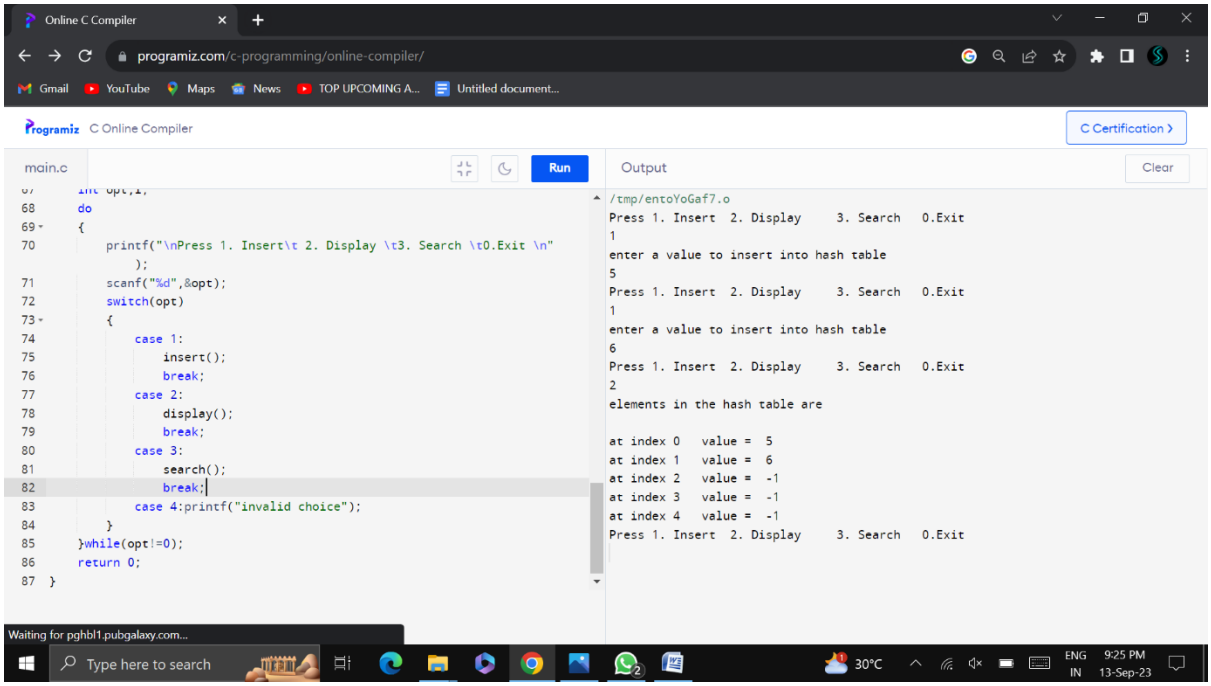

2. Write a C program to implement hashing using linear probing.

```
#include <stdio.h>

#include<stdlib.h>

#define s 5


int h[s];


void insert()

{


 int key,index,i,hkey;

 printf("\nenter a value to insert into hash table\n");

 scanf("%d",&key);

 hkey=key%s;

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

   {
```

```
    index=(hkey+i)%s;


    if(h[index] == -1)
    {
       h[index]=key;
        break;
    }


   }


   if(i == s)


    printf("\nelement cannot be inserted\n");
}
void search()
{


 int key,index,i,hkey;
 printf("\nenter search element\n");
 scanf("%d",&key);
 hkey=key%s;
 for(i=0;i<s; i++)
 {
   index=(hkey+i)%s;
   if(h[index]==key)
   {
```

```
      printf("value is found at index %d",index);
      break;
    }
  }
  if(i == s)
    printf("\n value is not found\n");
}
void display()
{

  int i;

  printf("\nelements in the hash table are \n");

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

  printf("\nat index %d \t value =  %d",i,h[i]);

}
 int main()
{
            int j;
            for(j=0;j<s;j++)
            h[j]=-1;
    int opt,i;
    do
    {
```

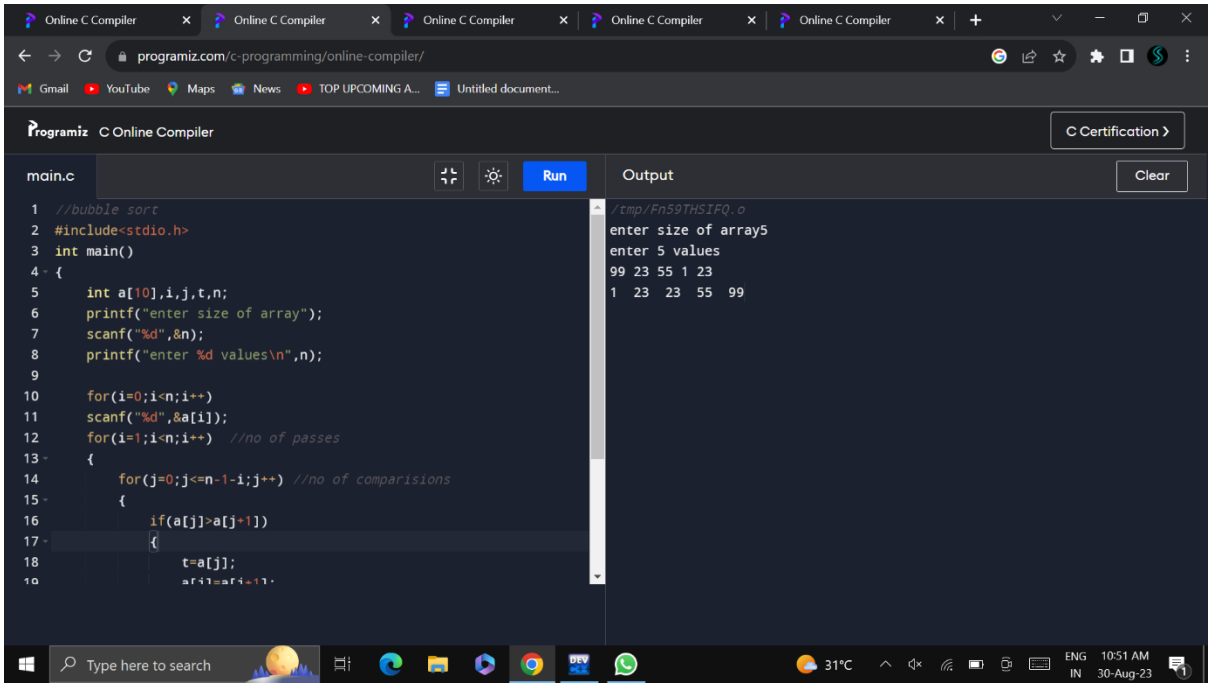
```
        printf("\nPress 1. Insert\t 2. Display \t3. Search \t0.Exit \n");
        scanf("%d",&opt);
        switch(opt)
        {
           case 1:
              insert();
              break;
           case 2:
              display();
              break;
           case 3:
              search();
              break;
           case 4:printf("invalid choice");
        }
     }while(opt!=0);
     return 0;
}
```




3. Write a C program to implement bubble sort.

```
#include<stdio.h>

int main()

{

        int a[10],i,j,t,n;

        printf("enter size of array");

        scanf("%d",&n);

        printf("enter %d values\n",n);


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

        scanf("%d",&a[i]);

        for(i=1;i<n;i++)  //no of passes

        {

                for(j=0;j<=n-1-i;j++) //no of comparisions

                {

                        if(a[j]>a[j+1])

                        {
```

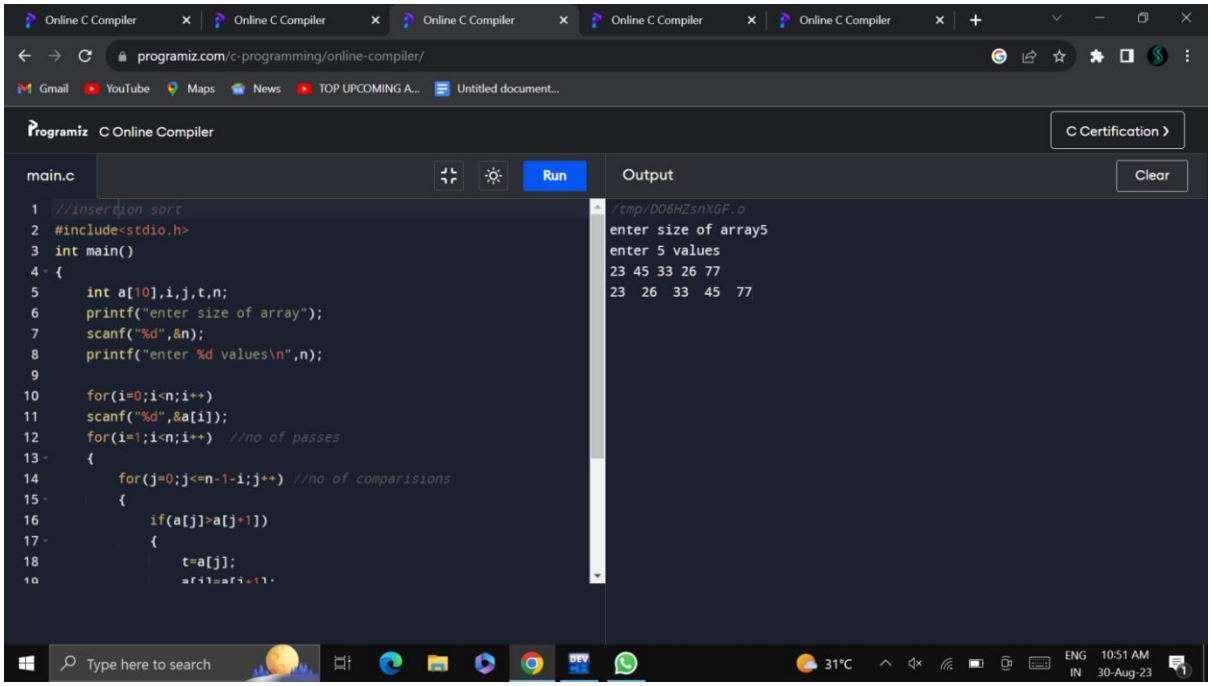
```
                    t=a[j];
                    a[j]=a[j+1];
                    a[j+1]=t;
                }
            }

        }
        for(i=0;i<n;i++)
            printf("%4d",a[i]);
}
```




4. Write a C program to implement insertion sort.

```
#include<stdio.h>
int main()
{
        int a[10],i,j,t,n;
        printf("enter size of array");
        scanf("%d",&n);
```

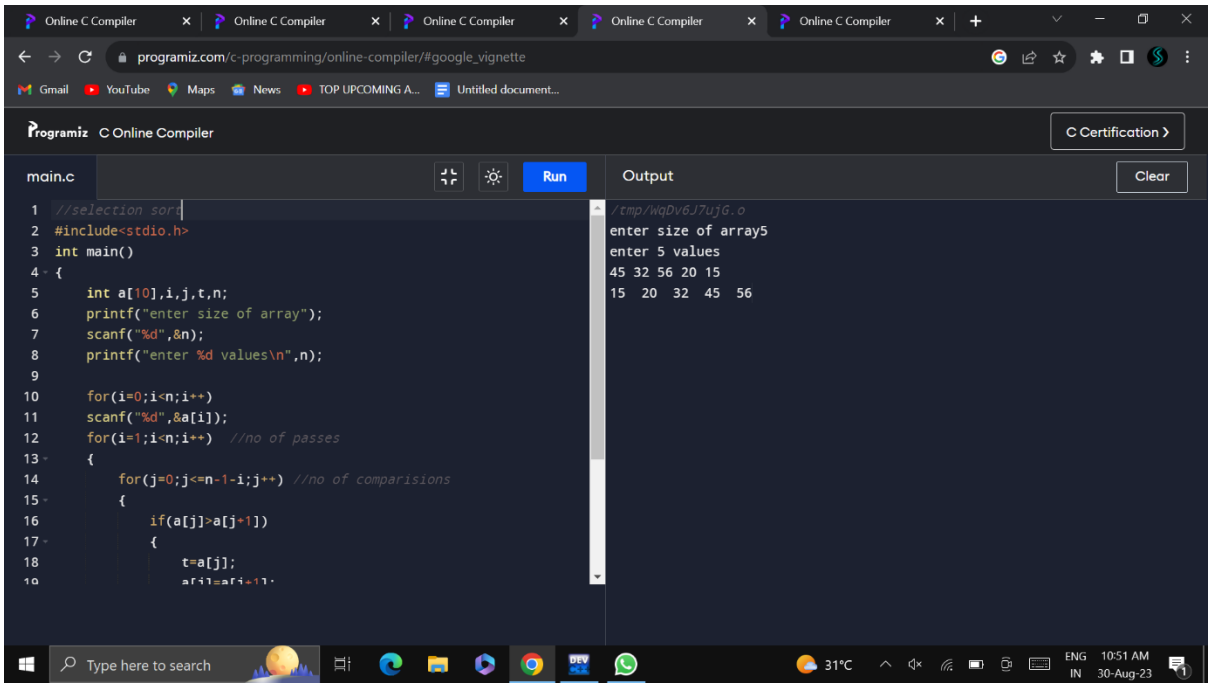
```
	printf("enter %d values\n",n);


	for(i=0;i<n;i++)
	scanf("%d",&a[i]);
	for(i=1;i<n;i++)
	{
		t=a[i];
		j=i-1;
		while(j>=0 && a[j]>t) //dack traversal
		{
			a[j+1]=a[j];
			j--;
		}
		a[j+1]=t;
	}
	for(i=0;i<n;i++)
		printf("%4d",a[i]);
}
```




5. Write a c program to implement selection sort.

```
#include<stdio.h>

int main()

{

        int a[10],i,j,min,n,t;

        printf("enter size of array");

        scanf("%d",&n);

        printf("enter %d values\n",n);


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

        scanf("%d",&a[i]);

        for(i=0;i<n-1;i++)

        {

                min=i;

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

                {
```

```
                if(a[j]<a[min])
                {
                        min=j;
                }
        }
        if(min!=i)
        {
                t=a[i];
                a[i]=a[min];
                a[min]=t;
        }

    }
    for(i=0;i<n;i++)
            printf("%4d",a[i]);
}
```




6. Write a c program to implement Quick sort.

```
#include<stdio.h>
int a,ub,lb;
void swap(int *x,int *y)
{
        int t;
        t=*x;
        *x=*y;
        *y=t;

}

 int partition(int a[],int lb,int ub)
{
        int pivot;
        int s,e;
        pivot=a[lb];
        s=lb;
        e=ub;
        while(s<e)
        {
        while(a[s]<=pivot)
        {
        s++;
   }
        while(a[e]>pivot)
        {
        e--;
```

```
    }
        if(s<e)
        {
            swap(&a[s],&a[e]);
        }
   }
   swap(&a[lb],&a[e]);
        return e;
}

void quicksort(int a[],int lb,int ub)
{
        if(lb<ub)
        {
            int l=partition(a,lb,ub);
            quicksort(a,lb,l-1);
            quicksort(a,l+1,ub);
        }
}
int main()
{
        int i, a[50],n,lb,ub;
        printf("enter size of array");
        scanf("%d",&n);
        printf("enter elements\n");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
```

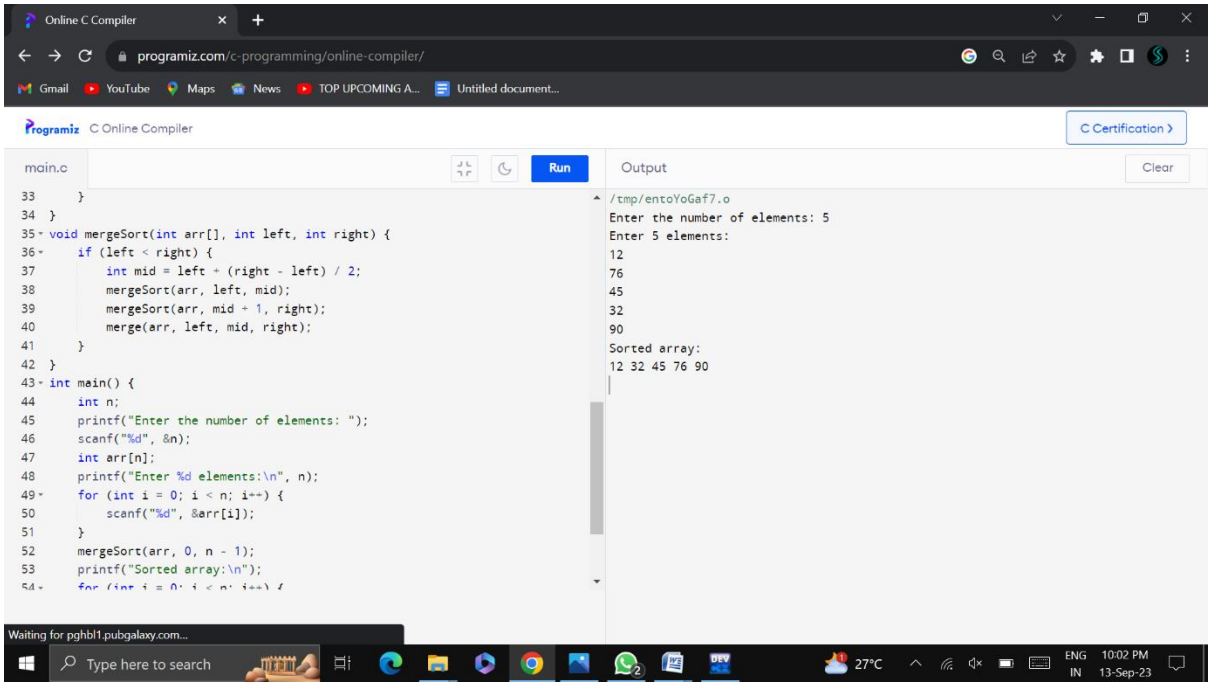
```
    lb=0,ub=n-1;
    quicksort(a,lb,ub);
    for(i=0;i<n;i++)
    printf("%d  ",a[i]);


}
```


7. Write a C program to implement Merge sort

```
#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
```

```
        R[j] = arr[mid + 1 + j];
    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
```

```
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    mergeSort(arr, 0, n - 1);
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```


Online C Compiler
programiz.com/c-programming/online-compiler/
Gmail
YouTube
Maps
News
TOP UPCOMING A...
Untitled document...
Programiz C Online Compiler
C Certification
main.c
Run
Output
Clear
}
}
void mergeSort(int arr[], int left, int right) {
if (left < right) {
int mid = left + (right - left) / 2;
mergeSort(arr, left, mid);
mergeSort(arr, mid + 1, right);
merge(arr, left, mid, right);
}
}
int main() {
int n;
printf("Enter the number of elements: ");
scanf("%d", &n);
int arr[n];
printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
mergeSort(arr, 0, n - 1);
printf("Sorted array:\n");
/tmp/entoYoGaf7.o
Enter the number of elements: 5
Enter 5 elements:
12
76
45
32
90
Sorted array:
12 32 45 76 90
Waiting for pghbl1.pubgalaxy.com...
Type here to search
27°C
ENG
IN
10:02 PM
13-Sep-23