**DATA STRUCTURES**

**DAY-4**

**IMPLEMENTATION OF 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;

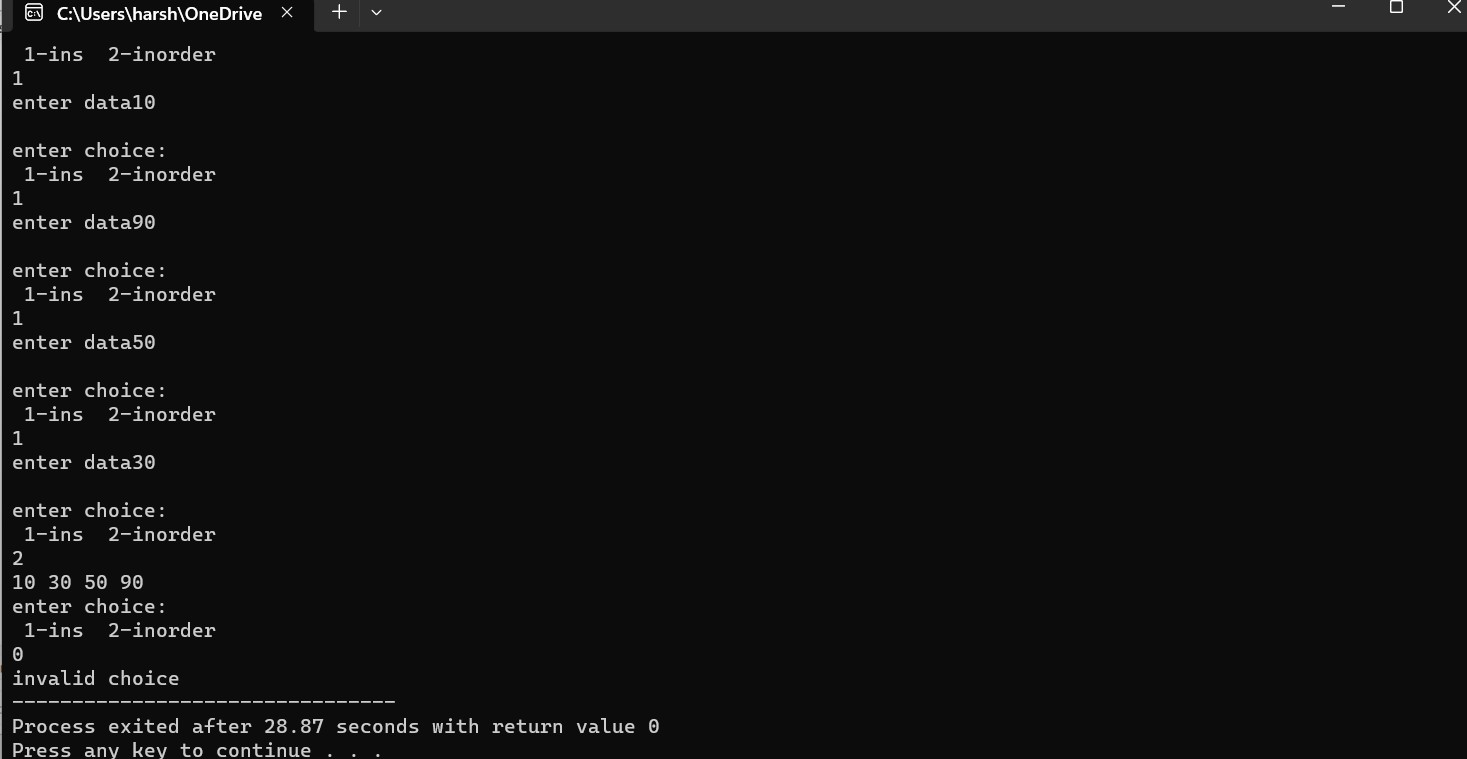
default: printf("invalid choice");

}

}

while (choice!=0);

}



**IMPEMENTATION OF 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;

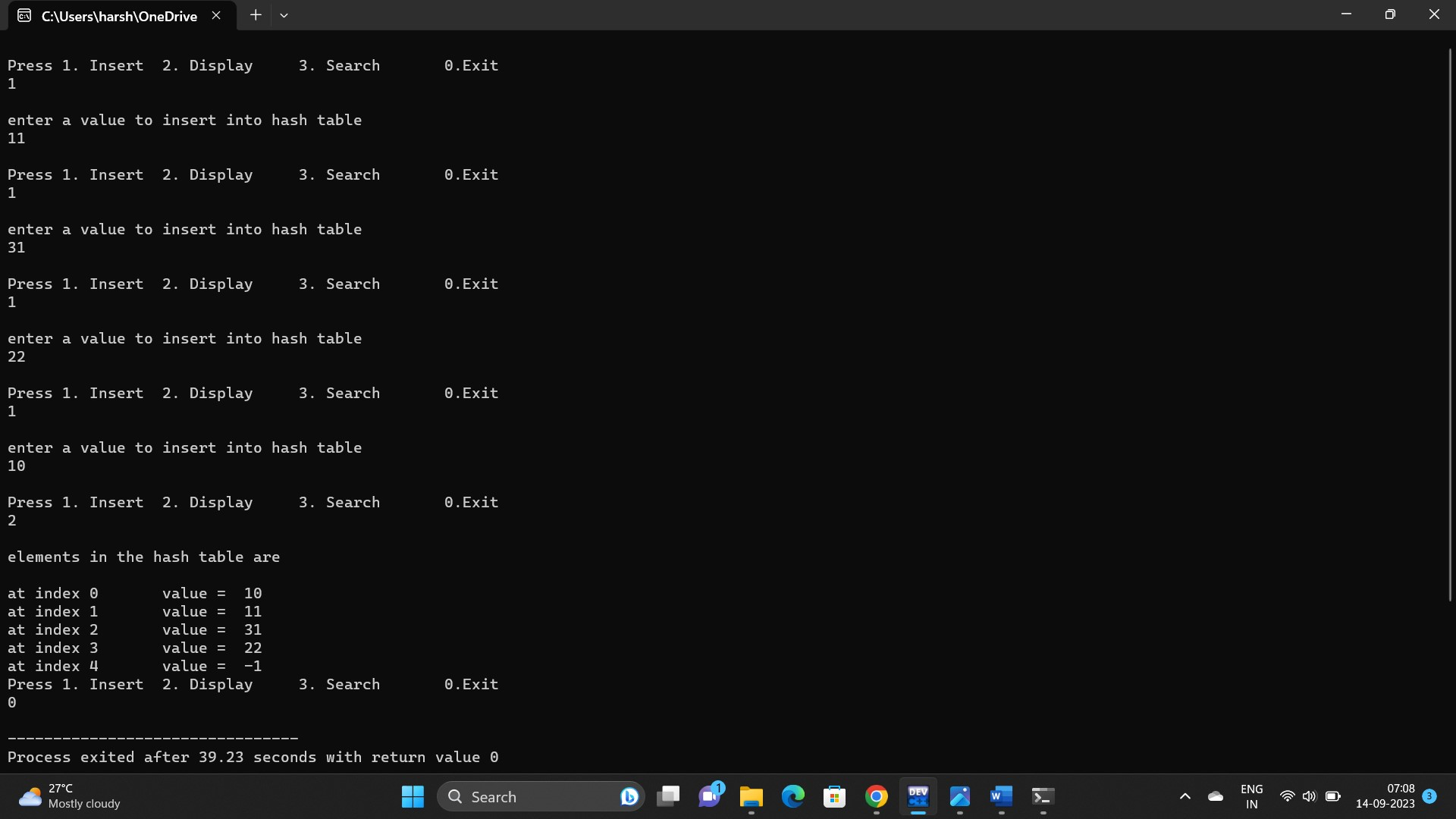
case 4:printf("invalid choice");

}

}while(opt!=0);

return 0;

}



**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;

}

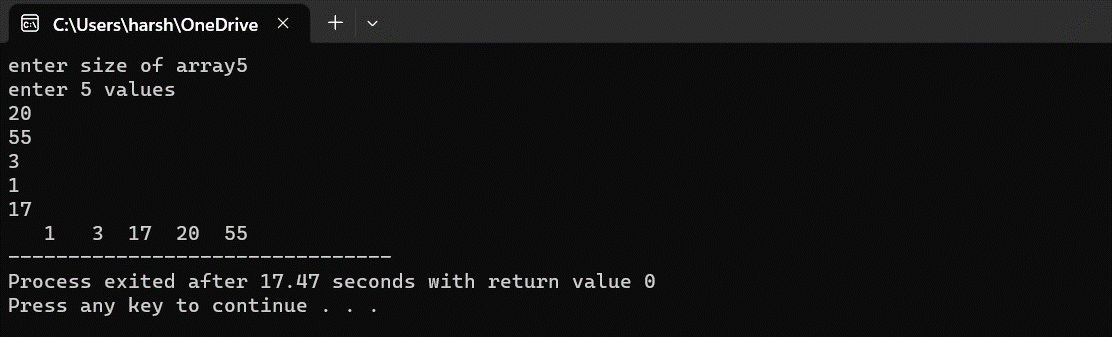
}

}

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

printf("%4d",a[i]);

}



**INSERSION 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--;

}

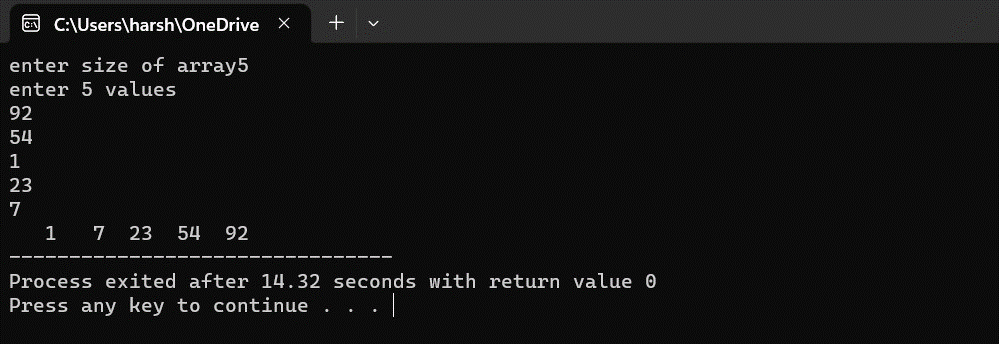
a[j+1]=t;

}

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

printf("%4d",a[i]);

}



**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;

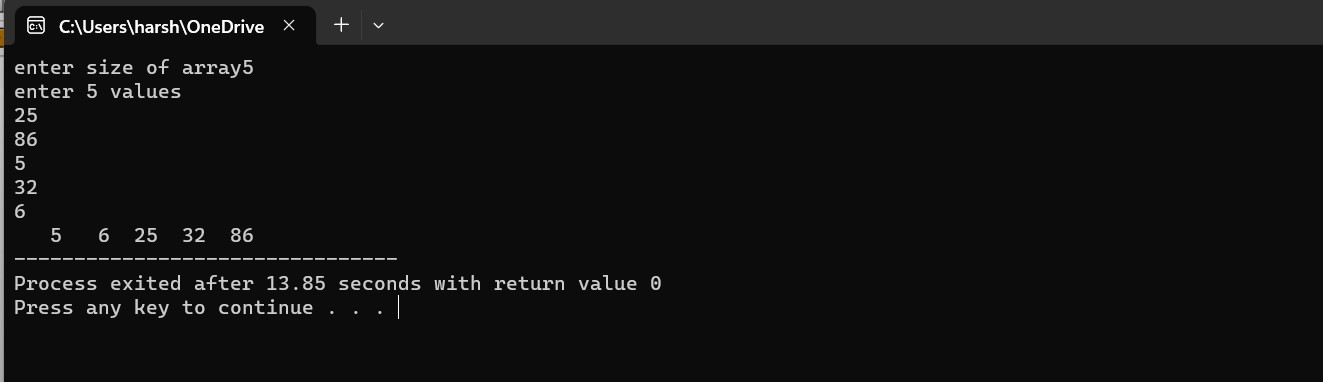
}

}

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

printf("%4d",a[i]);

}



**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]);

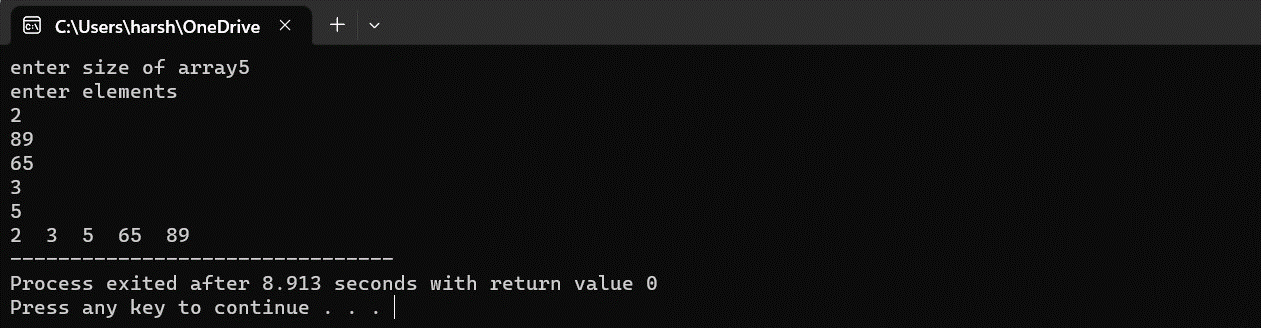
lb=0,ub=n-1;

quicksort(a,lb,ub);

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

printf("%d ",a[i]);

}



**MERGER SORT**

#include<stdio.h>

void merge(int arr[],int l, int mid, int r)

{

int n=mid+1-l;

int m=r-mid;

int a[n], b[m],i,j;

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

{

a[i]=arr[l+i];

}

for(j=0;j<m;j++)

{

b[j]=arr[mid+1+j];

}

i=0;j=0;int k=l;

while(i<n && j<m)

{

if(a[i]<b[j])

{

arr[k]=a[i];

k++ ;

i++;

}

else

{

arr[k]=b[j];

k++;

j++;

}

}

while(i<n)

{

arr[k]=a[i];

k++;

i++;

}

while(j<m)

{

arr[k]=b[j];

k++;

j++;

}

}

mergesort(int arr[],int l, int r)

{

int mid;

if(l<r)

{

mid=(l+r)/2;

mergesort(arr,l,mid);

mergesort(arr,mid+1,r);

merge(arr,l,mid,r);

}

}

int main()

{

int arr[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",&arr[i]);

mergesort(arr,0,n-1);

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

printf("%4d",arr[i]);

}

