                           DSP LAB – Assignment - 8

                             19-nov-2021

Q:1 Design a HashTable (atleast size >= 10) for Integers

        (Use Collision Resolution Technique : -Quadratic Probing)

        Provide following Function

1 Insert a Key

2. Delete a Key

          3. Search a Key Using key Value

4. Calculate Load Factor

Source Code:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int no\_of\_Values = 0;

int \*arr=NULL;

void insert(int n)

{

int pvalue,value,key,i,hkey;

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

scanf("%d",&value);

pvalue = value;

hkey=abs(value)%n;

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

{

key=(hkey+(i\*i))%n;

if(arr[key] == INT\_MAX && key == 0)

{

arr[key]=pvalue;

no\_of\_Values = no\_of\_Values + 1;

break;

}

if(arr[key] == INT\_MAX && key!= 0)

{

arr[key]=pvalue;

no\_of\_Values = no\_of\_Values + 1;

break;

}

}

if(i == n)

printf("\nValue cannot be inserted - HashTable was filled\n");

}

void search(int n)

{

int value,key,i,hkey;

printf("\nEnter Search Value:\n");

scanf("%d",&value);

hkey=value%n;

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

{

key=(hkey+(i\*i))%n;

if(arr[key]==value)

{

printf("Value is found at index %d",key);

break;

}

}

if(i == n)

printf("\n value is not found\n");

}

void h\_delete(int n)

{

int value,key,i,hkey;

printf("\nEnter Value to delete:\n");

scanf("%d",&value);

hkey=value%n;

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

{

key=(hkey+(i\*i))%n;

if(arr[key]==value)

{

arr[key] = INT\_MAX;

no\_of\_Values = no\_of\_Values - 1;

break;

}

}

if(i == n)

printf("\n value is not found\n");

}

void display(int n)

{

int i;

printf("\nElements in the Hash table are :\n");

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

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

}

int main()

{

int n,i,choice;

float Loadfactor;

printf("Enter Size of an Array:");

scanf("%d",&n);

if(n<10)

{

printf("\nPlease Enter Atleast Size of an Array Greater than or Equal to 10: ");

scanf("%d",&n);

}

if(n<10)

{

printf("\nPlease Enter Atleast Size of an Array Greater than or Equal to 10");

exit(0);

}

arr=(int\*)malloc(n\*sizeof(int));

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

{

arr[i]= INT\_MAX;

}

while(choice!=5)

{

printf("\n Using Collision Resolution Technique : -Quadratic Probing\n 1. Inserting elements in the hash table\n 2. Searching elements from the hash table\n 3. Deleting an element from the hash table\n 4. Calculate the load factor of the hash table\n 5. Exit\n Enter Your Choice: ");

if(scanf("%d",&choice)==1)

{

if(choice==1)

{

insert(n);

display(n);

}

else if(choice==2)

{

search(n);

}

else if(choice==3)

{

h\_delete(n);

display(n);

}

else if(choice==4)

{

Loadfactor = (float)no\_of\_Values/n;

display(n);

printf("\nno\_of\_Values:%d",no\_of\_Values);

printf("\nLoad Factor of the Hash table : %f ",Loadfactor);

}

else if(choice==5)

{

exit(0);

}

else{

printf("Invalid choice!");

}

}

else{

printf("Enter Numbers from 1 to 5");

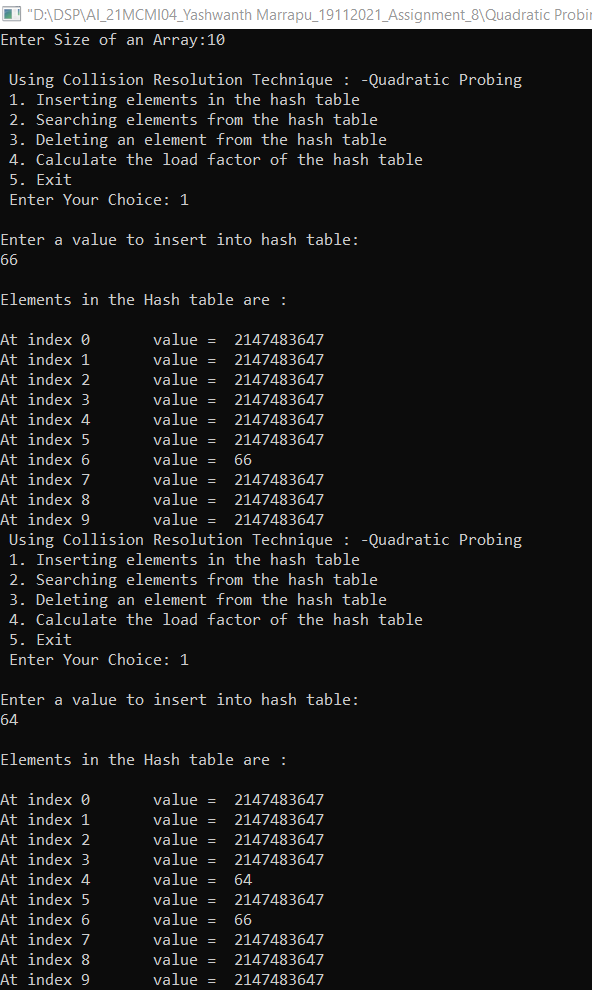
exit(0);

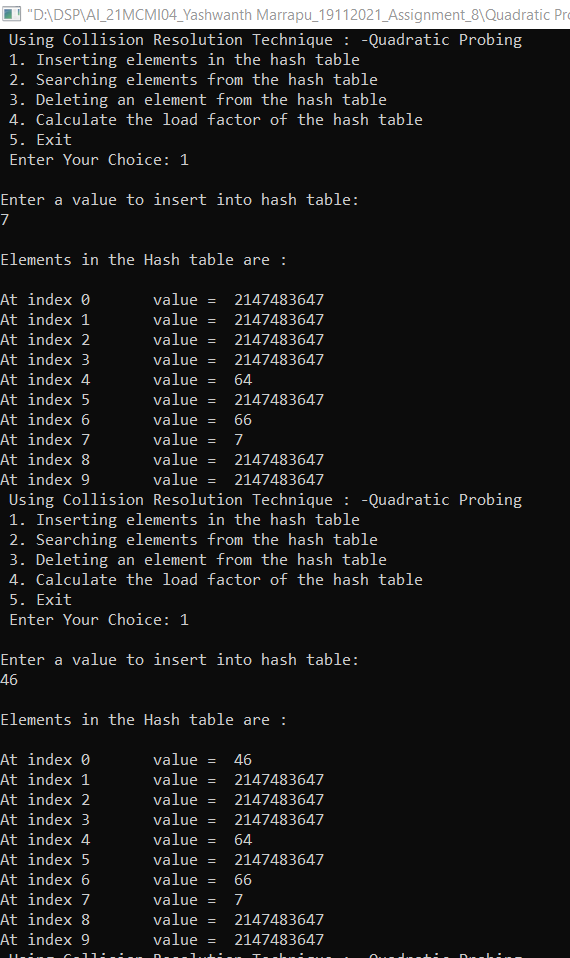
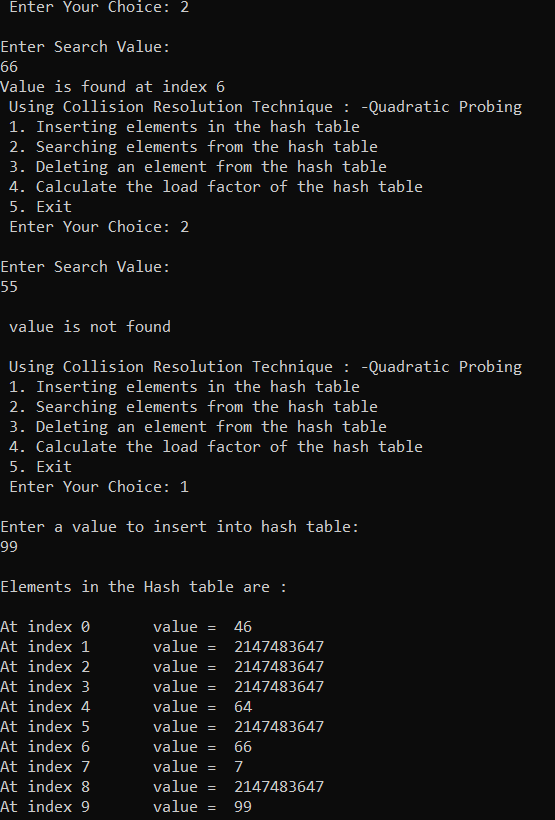
}

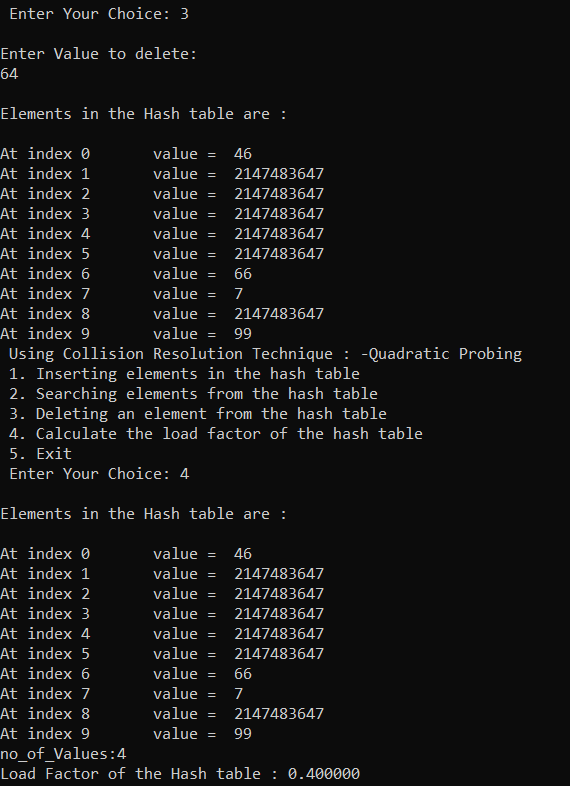
}

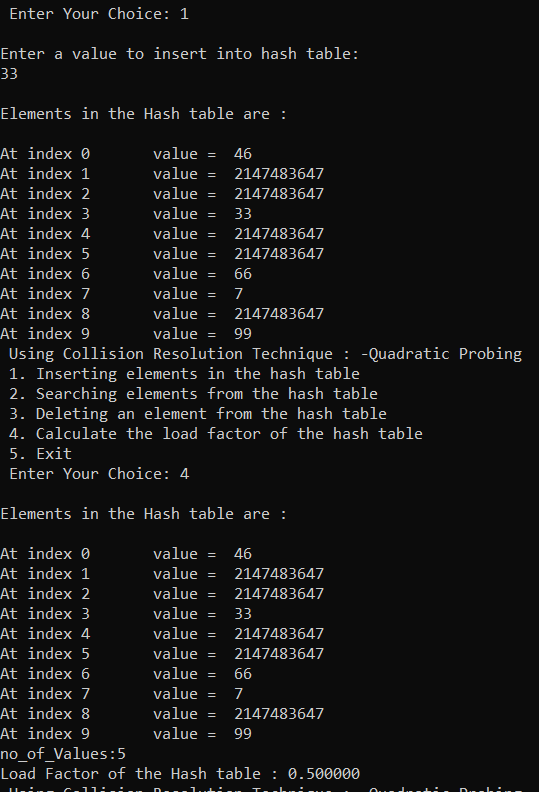
}

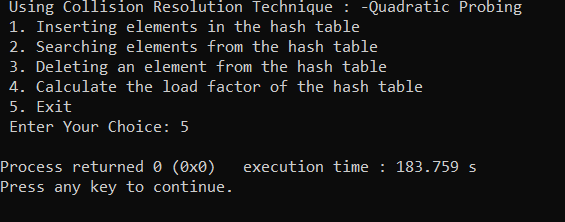
Output:









Q:2 Design a HashTable (atleast size >= 10) for Integers

        (Use Collision Resolution Technique : -Double Hashing)

        Provide following Function

1 Insert a Key

2. Delete a Key

          3. Search a Key Using key Value

4. Calculate Load Factor

Source Code:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int no\_of\_Values = 0;

int \*arr=NULL;

void insert(int n)

{

int pvalue,value,key,i,hkey,Double\_hash;

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

scanf("%d",&value);

Double\_hash = 7-(value %7);

pvalue = value;

hkey=abs(value)%n;

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

{

key=(hkey+(i\*Double\_hash))%n;

if(arr[key] == INT\_MAX && key == 0)

{

arr[key]=pvalue;

no\_of\_Values = no\_of\_Values + 1;

break;

}

if(arr[key] == INT\_MAX && key!= 0)

{

arr[key]=pvalue;

no\_of\_Values = no\_of\_Values + 1;

break;

}

}

if(i == n)

printf("\nValue cannot be inserted - HashTable was filled\n");

}

void search(int n)

{

int value,key,i,hkey,Double\_hash;

printf("\nEnter Search Value:\n");

scanf("%d",&value);

Double\_hash = 7-(value %7);

hkey=value%n;

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

{

key=(hkey+(i\*Double\_hash))%n;

if(arr[key]==value)

{

printf("Value is found at index %d",key);

break;

}

}

if(i == n)

printf("\n value is not found\n");

}

void h\_delete(int n)

{

int value,key,i,hkey,Double\_hash;

printf("\nEnter Value to delete:\n");

scanf("%d",&value);

Double\_hash = 7-(value %7);

hkey=value%n;

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

{

key=(hkey+(i\*Double\_hash))%n;

if(arr[key]==value)

{

arr[key] = INT\_MAX;

no\_of\_Values = no\_of\_Values - 1;

break;

}

}

if(i == n)

printf("\n value is not found\n");

}

void display(int n)

{

int i;

printf("\nElements in the Hash table are :\n");

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

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

}

int main()

{

int n,i,choice;

float Loadfactor;

printf("Enter Size of an Array:");

scanf("%d",&n);

if(n<10)

{

printf("\nPlease Enter Atleast Size of an Array Greater than or Equal to 10: ");

scanf("%d",&n);

}

if(n<10)

{

printf("\nPlease Enter Atleast Size of an Array Greater than or Equal to 10");

exit(0);

}

arr=(int\*)malloc(n\*sizeof(int));

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

{

arr[i]= INT\_MAX;

}

while(choice!=5)

{

printf("\n Using Collision Resolution Technique : -Double Hashing\n 1. Inserting elements in the hash table\n 2. Searching elements from the hash table\n 3. Deleting an element from the hash table\n 4. Calculate the load factor of the hash table\n 5. Exit\n Enter Your Choice: ");

if(scanf("%d",&choice)==1)

{

if(choice==1)

{

insert(n);

display(n);

}

else if(choice==2)

{

search(n);

}

else if(choice==3)

{

h\_delete(n);

display(n);

}

else if(choice==4)

{

Loadfactor = (float)no\_of\_Values/n;

display(n);

printf("\nno\_of\_Values:%d\n",no\_of\_Values);

printf("\nLoad Factor of the Hash table : %f\n ",Loadfactor);

}

else if(choice==5)

{

exit(0);

}

else{

printf("Invalid choice!");

}

}

else{

printf("Enter Numbers from 1 to 5");

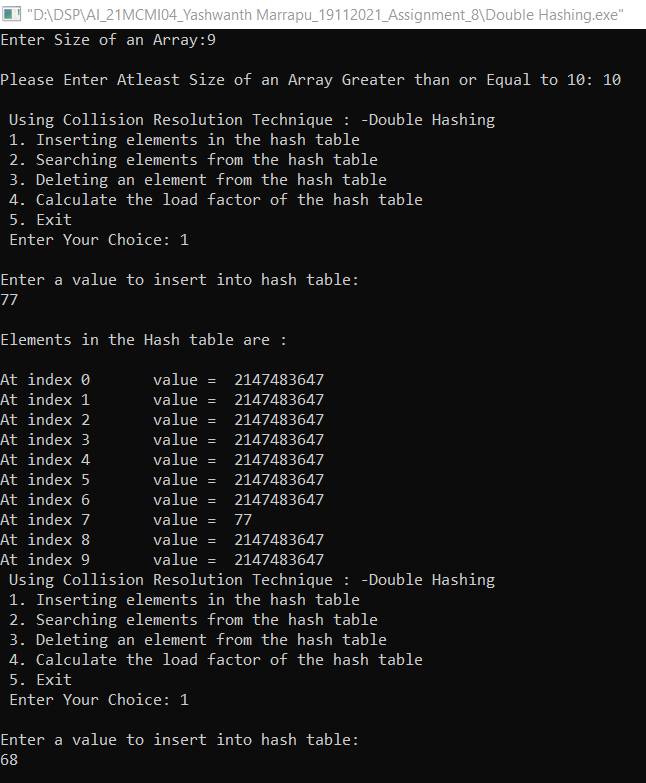
exit(0);

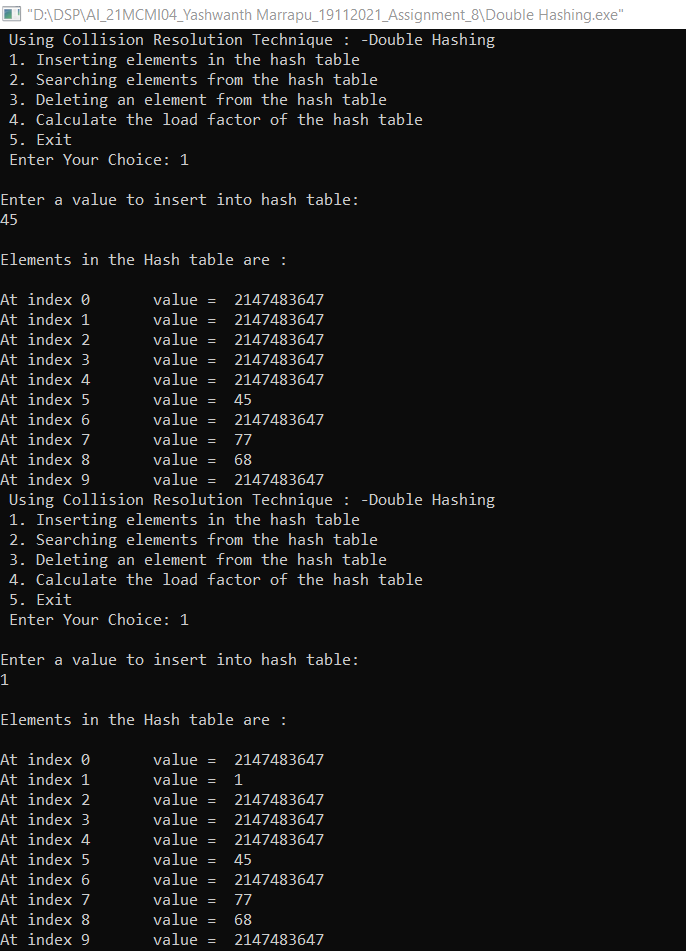
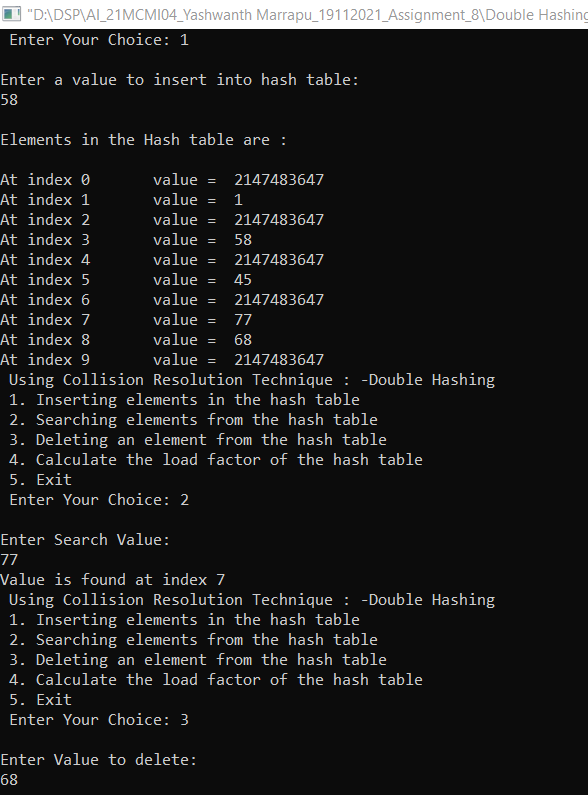
}

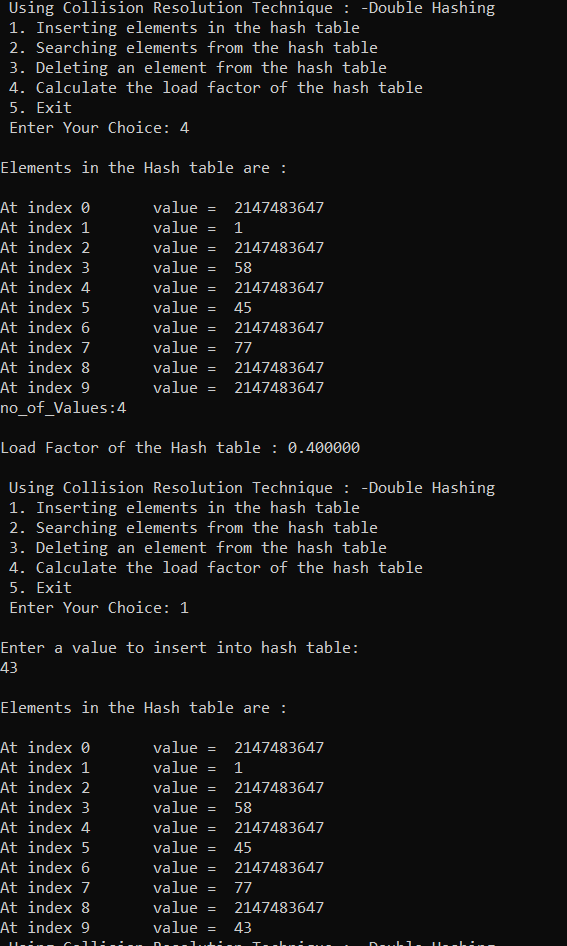
}

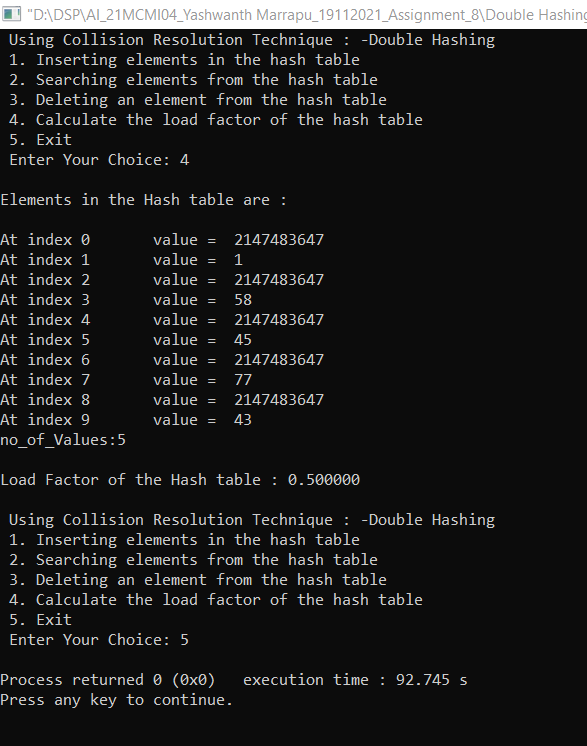
}

Output:







Q:3 Implement Heap-sort Algorithm Using Min-Heapify and Max-Heapify

       Procedures.

       Your Program Should Work Like This

       Choose option

1. Using min-heapify
2. Using Max-heapify

After selecting either of the option

     Implement following functions(for both options) :

1. Build Heap by inserting some integers.
2. Insert a key into heap.
3. Delete function.
4. Sort function.

Source Code:

//heap sort program

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define MAX 100

int arr[MAX]={0};

int flag=0;

int count=0;

void swap(int \*x, int \*y)

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

void Heapify\_Min(int arr[], int n, int i)

{

int min = i;

int l = (2 \* i) + 1;

int r = (2 \* i)+ 2;

if (l < n && arr[l] < arr[min])

{

min = l;

}

if (r < n && arr[r] < arr[min])

{

min = r;

}

if (min != i)

{

swap(&arr[i], &arr[min]);

Heapify\_Min(arr, n, min);

}

}

void Heapify\_Max(int arr[], int n, int i)

{

int max = i;

int l = ( 2 \* i) + 1;

int r = (2 \* i )+ 2;

if (l < n && arr[l] > arr[max])

{

max = l;

}

if (r < n && arr[r] > arr[max])

{

max = r;

}

if (max != i)

{

swap(&arr[i], &arr[max]);

Heapify\_Max(arr, n, max);

}

}

void display(int array[],int n)

{

int i;

printf("\n Array elements are: ");

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

{

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

}

printf("\n\n");

}

void Heapify()

{

int n,size,i;

printf("\n Enter Size of an Array : ");

scanf("%d",&n);

count = n;

printf("\n Enter Elements in to Array: ");

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

{

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

}

display(arr,n);

if(flag==1)

{

//Max heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Max(arr, n, i);

}

printf("\n Max heap build\n");

}

else if (flag == 2)

{

//Min heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Min(arr, n, i);

}

printf("\n Min heap build\n");

}

}

void insert()

{

int key,i,j,n;

printf("\n Enter key value to insert in to heap: ");

scanf("%d",&key);

n=count;

arr[n]=key;

count++;

n++;

if(flag==1)

{

//Max heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Max(arr, n, i);

}

}

else if(flag==2)

{

//Min heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Min(arr, n, i);

}

}

printf("\n %d was inserted\n\n",key);

}

void heapsort()

{

int i,j,n;

n=count;

int arr2[n];

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

{

arr2[i]=arr[i];

}

// Heap sort

if(flag==1)

{

//Max heap

for (i = n - 1; i >= 0; i--)

{

swap(&arr2[0], &arr2[i]);

Heapify\_Max(arr2, i, 0);

}

}

else

{

//Min heap

for (i = n - 1; i >= 0; i--)

{

swap(&arr2[0], &arr2[i]);

// Heapify root element to get lowest element at root again

Heapify\_Min(arr2, i, 0);

}

}

display(arr2,n);

}

void del()

{

int i,j,key,n,index,k=0;

printf("\n Enter value to delete: ");

scanf("%d",&key);

n=count;

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

{

if(arr[i]==key)

{

k=1;

index=i;

break;

}

}

if(k==0)

{

printf("\n Element not found\n\n");

return;

}

swap(&arr[index],&arr[n-1]);

arr[n-1]=0;

count--;

n--;

if(flag==1)

{

//Max heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Max(arr, n, i);

}

}

else

{

//Min heap

for (i = n / 2 - 1; i >= 0; i--)

{

Heapify\_Min(arr, n, i);

}

}

printf("\n %d was deleted from heap\n\n",key);

}

void main()

{

int i,choice,choice1,check=0,val;

printf("\n Heap sort Algorithm \n");

printf("\n 1.Use Max heapify");

printf("\n 2.Use Min heapify");

printf("\n\n Enter Your choice: ");

val = scanf("%d",&choice);

if(val!=1)

{

printf("\n Invaid choice");

exit(0);

}

if(choice==1 || choice==2)

{

flag=choice;

while(1)

{

if(choice==1)

{

printf("\n\n Max heapify");

}

else if (choice == 2)

{

printf("\n\n Min heapify");

}

printf("\n 1. Build heap by inserting Integers");

printf("\n 2. Insert a key in to heap ");

printf("\n 3. Delete function ");

printf("\n 4. Sort function");

printf("\n 5. Exit");

printf("\n\n Enter Your choice: ");

scanf("%d",&choice1);

if(choice1 == 1)

{

//build heap

if(check==0)

{

Heapify();

check=1;

}

else

{

printf("\n Heap already build\n");

}

}

else if(choice1 == 2)

{

//insert a key

if(check==0)

{

printf("\n Heap Was not build \n\n");

}

else

{

insert();

}

}

else if(choice1==3)

{

//delete

if(check==0)

{

printf("\n Heap Was not build\n\n");

}

else

{

del();

}

}

else if(choice1==4)

{

//sort function

if(check==0)

{

printf("\n Heap was not build\n\n");

}

else

{

heapsort();

}

}

else if(choice1==5)

{

break;

}

else

{

printf("\n Invalid input. Try again!");

}

}

}

else

{

printf("\n Invalid choice. !");

}

}

Output:

