NAME-YASH GANDHI SE IT BATCH A 14

Exp. 9. Heap

Implementation of a Min/Max Heap Structures.

1-Assume Sequence of numbers as input, build a min/max heap

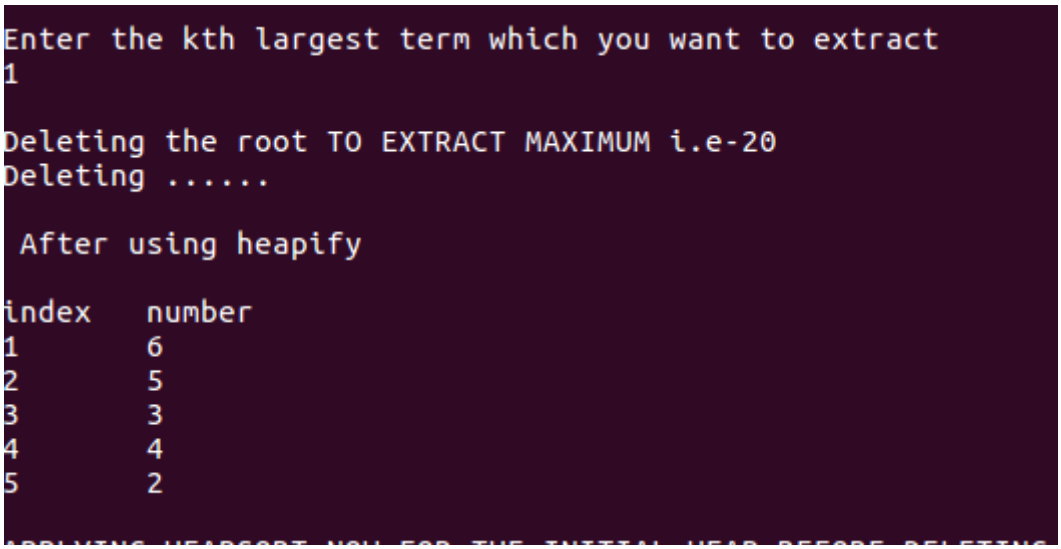
Max heap-

```
itlab@MM-Lab-410-U13:~/Desktop$ gcc heap.c
itlab@MM-Lab-410-U13:~/Desktop$ ./a.out
Enter the number of keys
5
Enter the data
4
Enter the data
5
Enter the data
3
Enter the data
6
Enter the data
2

 After using heapify

index   number
1       6
2       5
3       3
4       4
5       2
```

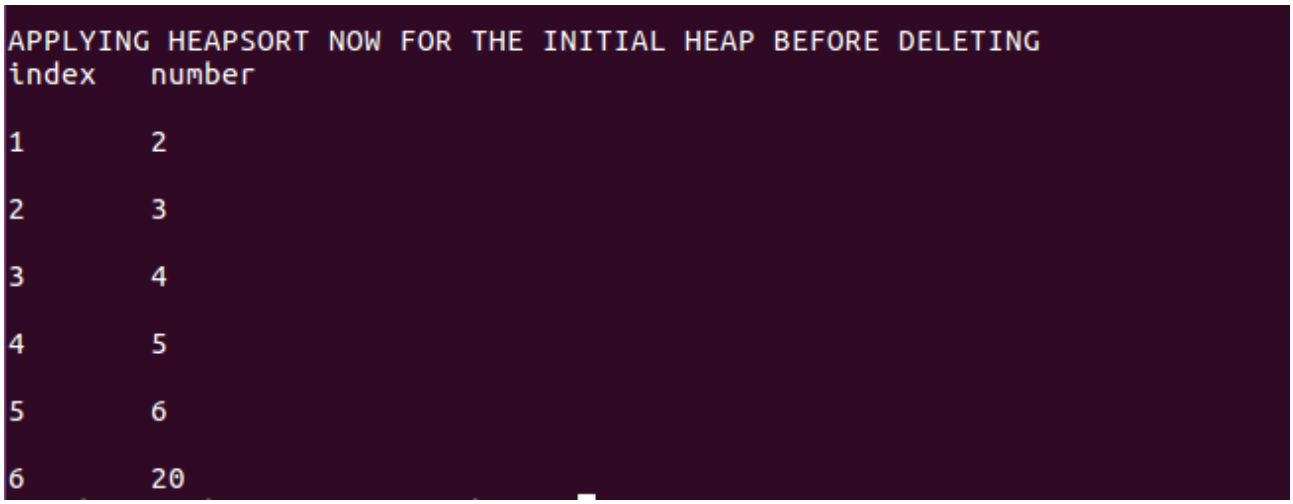
2- insert a node in heap . show the updated heap

```
Enter the data to insert
20

 After using heapify

index   number
1       20
2       5
3       6
4       4
5       2
6       3

Enter 1 to continue
0
```

3- Delete a node from heap. show the updated heap



4- Heapsort



PROGRAM-

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

void Max_heapify(int heap[],int i,int n)    //heapify function to maintain heap property
{
  int r=2*i+1;
  int l=2*i;
  int largest;
```

```
 if((l<=n)&&(heap[l]>heap[i]))
  {
    largest=l;
  }
 else
  {
     largest=i;
  }

   if((r<=n)&&(heap[r]>heap[largest]))
  {
    largest=r;
  }

  if(largest!=i)
   {
    int temp=heap[i];
    heap[i]=heap[largest];
    heap[largest]=temp;

     Max_heapify(heap,largest,n);
   }

 return;
}

void build_heap(int heap[],int n,int k)    //build heap function which calls heapify
{
 int i;

 for(i=n/2;i>=1;i--)
 {
   Max_heapify(heap,i,n);
 }

 if(k==1)
 {
 printf("\n After using heapify\n");
 printf("\nindex\tnumber\n");
  for(i=0;i<n;i++)
 {

   printf("%d\t%d\n",i+1,heap[i+1]);
 }
 }
}


int insert(int heap[],int n)              //insert function to insert a new value in the heap
{
int k,c=1,total=n;
```

```
int o=1;
while(c==1)
 {
  printf("\nEnter the data to insert\n");
   scanf("%d",&k);
  heap=(int*)realloc(heap,(total+1)*sizeof(int));

  heap[total+1]=k;
  total=total+1;

  build_heap(heap,total,o);  //maintain heap property after inserting

  printf("\nEnter 1 to continue\n");
  scanf("%d",&c);
 }

return total;

}

void delete(int heap[],int n)   //delete the root function
{
  int i,k,p;
  k=n;

   int v;
   printf("\nEnter the kth largest term which you want to extract\n");
  scanf("%d",&v);
  int o=1;
 for(i=0;i<v;i++)
 {
 int temp=heap[1];
 heap[1]=heap[k];
 heap[k]=temp;

  printf("\nDeleting the root TO EXTRACT MAXIMUM i.e-%d",temp);

  printf("\nDeleting ......\n");

 build_heap(heap,k-1,o);
 k=k-1;
 }

 o=0;
 build_heap(heap,n,o);   //to form heap again
  k=n;

 printf("\nAPPLYING HEAPSORT NOW FOR THE INITIAL HEAP BEFORE DELETING");

  for(i=0;i<n;i++)   //deleting all nodes and printing the array for heapsort
 {
 int temp=heap[1];
```

```
 heap[1]=heap[k];
 heap[k]=temp;
 build_heap(heap,k-1,o);
 k=k-1;
 }

  printf("\nindex\tnumber\n");

 for(i=0;i<n;i++)
 {
   printf("\n%d\t%d\n",i+1,heap[i+1]);
 }

}


void main()
{
 int *heap;

 int n,i,k;
 int x=1;
 printf("Enter the number of keys\n");
 scanf("%d",&n);
 heap=(int*)malloc(n*sizeof(int));            //array dynamic memory allocation
 for(i=0;i<n;i++)
 {
   printf("Enter the data\n");
   scanf("%d",&k);
   heap[i+1]=k;
 }


 build_heap(heap,n,x);                          //function calls

 n=insert(heap,n);

 delete(heap,n);

 return;
}
```

Output-

```
itlab@MM-Lab-410-U13:~/Desktop$ gcc heap.c
itlab@MM-Lab-410-U13:~/Desktop$ ./a.out
Enter the number of keys
5
Enter the data
4
```

Enter the data
5
Enter the data
3
Enter the data
6
Enter the data
2

After using heapify

index number
1 6
2 5
3 3
4 4
5 2

Enter the data to insert
20

After using heapify

index number
1 20
2 5
3 6
4 4
5 2
6 3

Enter 1 to continue
0

Enter the kth largest term which you want to extract
1

Deleting the root TO EXTRACT MAXIMUM i.e-20
Deleting .....

After using heapify

index number
1 6
2 5
3 3
4 4
5 2

APPLYING HEAPSORT NOW FOR THE INITIAL HEAP BEFORE DELETING
index number

1	2
2	3
3	4
4	5
5	6
6	20