

4
 3
 2
 1
 Enter your choice: 3

Result

Thus rotations were performed as a result of insertions to AVL Tree.

Ex. No. 9 **Binary Heap**

Date:

Aim

To build a binary heap.

Algorithm

1. Start
2. In a heap, for every node X with parent p , the key in p is smaller than or equal to the key in X .
3. For insertion operation
 - a. Add the element to the bottom level of the heap.
 - b. Compare the added element with its parent; if they are in the correct order, stop.
 - c. If not, swap the element with its parent and return to the previous step.
4. For deleteMin operation
 - a. Replace the root of the heap with the last element on the last level.
 - b. Compare the new root with its children; if they are in the correct order, stop.
 - c. If not, Swap with its smaller child in a min-heap
5. Stop

/ Priority Queue using Heap */*

```
#include<stdio.h>
#include<math.h>
#define MAX 100
```

```
void swap(int*, int*);
void display(int[],int);
```

```

void insert(int[],int,int,int);
intdel_hi_priori(int[],int,int);

int main()
{
    intlb,choice,num,n,a[MAX],data,s;
    choice = 0;
    n=0; //Represents number of nodes in the queue
    lb=0; //Lower bound of the array is initialized to 0
    while(choice != 4)
    {
        printf(".....MAIN MENU.....\n");
        printf("\n1.Insert.\n");
        printf("2.Delete.\n");
        printf("3.Display.\n");
        printf("4.Quit.\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter data to be inserted : ");
                scanf("%d",&data);
                insert(a,n,data,lb);
                n++;
                break;
            case 2:
                s=del_hi_priori(a,n+1,lb);
                if(s!=0)
                    printf("The deleted value is : %d",s);
                if(n>0)
                    n--;
                break;
            case 3:
                printf("\n");
                display(a,n);
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
        printf("\n\n");
    }
    return 0;
}

//-----INSERT-----
void insert(int a[],intheapsize,intdata,intlb)
{
    inti,p;
    int parent(int);
    if(heapsize==MAX)

```

```

    {
        printf("Queue Is Full!!\n");
        return;
    }
    i=lb+heapsize;
    a[i]=data;
    while(i>lb&& a[p=parent(i)]<a[i])
    {
        swap(&a[p],&a[i]);
        i=p;
    }
}

//-----DELETE-----
intdel_hi_priori(int a[],intheapsize,intlb)
{
    intdata,i,l,r,max_child,t;
    int left(int);
    int right(int);
    if(heapsize==1)
    {
        printf("Queue Is Empty!!\n");
        return 0;
    }
    t=a[lb];
    swap(&a[lb],&a[heapsize-1]);
    i=lb;
    heapsize--;
    while(1)
    {
        if((l=left(i))>=heapsize)
            break;
        if((r=right(i))>=heapsize)
            max_child=l;
        else
            max_child=(a[l]>a[r])?l:r;
        if(a[i]>=a[max_child])
            break;
        swap(&a[i],&a[max_child]);
        i=max_child;
    }
    return t;
}

//Returns Parent Index
int parent(int i)
{
    float p;
    p=((float)i/2.0)-1.0;
    return ceil(p);
}

//Return Leftchild Index

```

```
int left(int i)
{
    return 2*i+1;
}

//Return Rightchild Index
int right(int i)
{
    return 2*i+2;
}

//-----DISPLAY-----
void display(int a[],int n)
{
    int i;
    if(n==0)
    {
        printf("Queue Is Empty!!\n");
        return;
    }
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\n");
}

//-----SWAP-----
void swap(int*p,int*q)
{
    int temp;
    temp=*p;
    *p=*q;
    *q=temp;
}
```