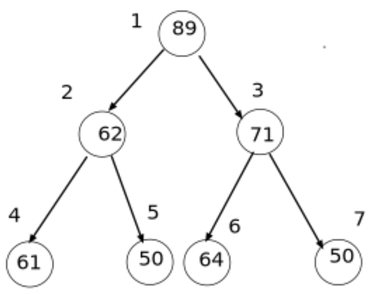
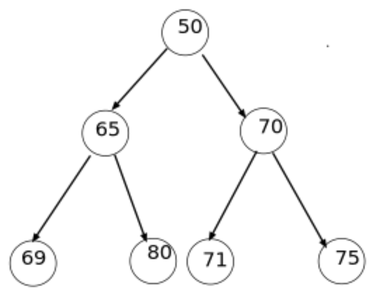
Binary Heap



Max Binary Heap

Complete Binary Tree



Min Binary Heap

23 12 19 35 33 42 25

```
struct BinaryHeap
{
    int *heap;
    int size;
   int length;
};
```

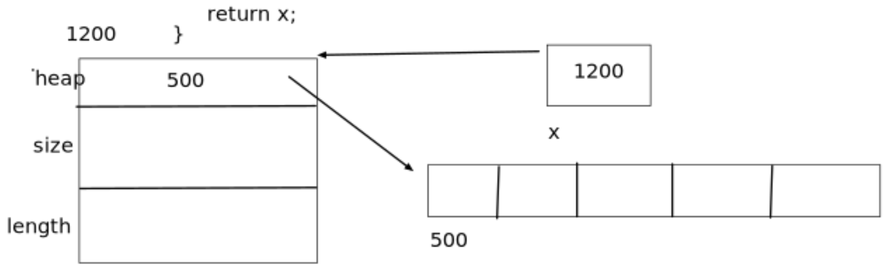
```
struct BinaryHeap* BinaryHeap(int n)
{
    struct BinaryHeap* x = (struct BinaryHeap*)malloc(sizeof(struct BinaryHeap));
    x -> size = n;
    x-> heap = (int*)malloc(sizeof(int)*x->size);
    x->length = 0;
    return x;
}
```

Data

1. int *heap
2. int size
3. int length

Operations

1. BinaryHeap(n)
2. isEmpty()
3. insert(x)
4. findMax() // findMin()
5. deleteMax() (extractMax()) // deleteMin() (extractMin())
6. size()




```
int isEmpty(struct BinaryHeap* x)
{
      if(x->length==0)
        {
            return 1;
        }
      return 0;
}
```

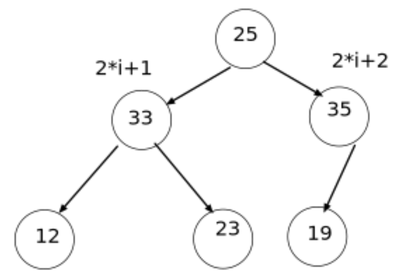
```
void insert(struct BinaryHeap* x, int y)
{
        int i = x->length;
        x->heap[i] = y;
        while(i!=0)
          {
            int k = (i-1)/2;
            if(x->heap[i] > x->heap[k])
               {
                    temp = x->heap[i];
                    x->heap[i] = x->heap[k];
                    x->heap[k] = x->heap[i];
                    i =k;
               }
             else
               {
                   break;
               }
          }
}
```

```
int findMax(struct BinaryHeap* x)
{
      if(isEmpty(x)==1)
              return -1;
      return x->heap[0];
}
```

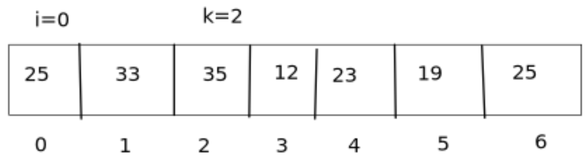
```
int extractMax(struct BinaryHeap* x)
{
      if(isEmpty(x)==1)
              return -1;
      int temp = x->heap[0],z;
      x->heap[0] = x->heap[x->length-1];
       x->length--;
      int i =0;

while(2*i+1 < x->length )
{

if(2*i+1 < x->length && 2*i+2 < x->length )
        {
              int k = x->heap[2*i+1] > x->heap[2*i+2]?2*i+1: 2*i+2;
              if(x->heap[i] < x->heap[k])
                 {
                    z = x->heap[i];
                    x->heap[i] = x->heap[k];
                    x->heap[k] = z;
                    i = k;
                 }
              else
                    break;
}
else if( 2*i+1 < x->length)
{
        int k = 2*i+1;
        if(x->heap[i] < x->heap[k])
                {
                    z = x->heap[i];
                    x->heap[i] = x->heap[k];
                    x->heap[k] = z;
                    i = k;
                }
              else
                    break;
}
}

return temp;
}
```

```
int size(struct BinaryHeap* x)
{
 return x->length;
}
```

23 12 19 35 33 42 25 i



x = 42



2*i+1

2*i+1

2*i+2

2*i+1 && 2*i+2

i = 0

2*i+1 = 1

2*i+2 = 2