ADS.

MANU.N.Y
1BM18CS053

Binomial Heap:-

```
Struct Node{
        int data, degree;
        Node *child, *sibling, *parent;
    }
Node * mergebinTree(Node * b1, Node * b2)
    if(b1 -> data > b2 -> data)
        swap (b1, b2)
    b2 -> parent = b1;
    b2 -> sibling = b1 -> child;
    b1 -> child = b2
    b1 -> degree += 1;
    return b1;

list  unionbinHeap (list l1, list l2)
        list <Node *> new, i1 = l1.begin(), i2 = l2.begin();
        while (i1 != l1.end() && i2 != l2.end())
            if ((i1) -> degree <= (i2) -> degree)
                new.pushback (i1)
                i1 ++
            else
                new.pushback (i2)
                i2 ++;
        while (i1 != l1.end())
            new.push_back (i1). i1++
        while (i2 != l2.end())
            new.pushback (i2). i2++
        return new;

list adjust (list heap)
    if(heap.size <= 1)
        return heap
    list <Node *> new, i1, i2, i3;
    i1 = i2 = i3 = heap.begin();
    if(heap.size == 2)
```

[illegible]

MANU . N . Y
1RM18LS053

```
        i2 = i1 ; i2++ ; i3 = heap.end();
    else
        i2++ ; i3 = i2 ; i3++ ;
    while (i1 != heap.end())
        if (i2 == heap.end())
            i1++;
        else if (i1 -> degree < (i2) -> degree)
        {   i1++ ; i2++;
            if (i3 ! = heap.end())
                i3++;
            else if (i3! = heap.end() && (i1) -> degree ==
                (i2) -> degree && (i1) -> degree == (i3) -> degree)
                i1++; i2++; i3++
            else if (i1 -> degree == i2 -> degree)
                i1 = mergebinTree (i1, i2)
                i2 = heap.erase (i2)
                if (i3 ! = heap.end())
                    i3++
        return heap
list insertATreeInHeap (list heap, Node Tree)
    list <Node *> temp
    temp.pushback (tree);
    temp = UnionBinHeap (heap, temp)
    adjust (heap);
    return heap;
list <Node *> insert ( list<Node*> head , int key)
    Node * temp = new Node (key);
    return insertATreeInHeap (head, temp);
Node getMin (list<Node *> heap)
    list <Node*> :: iterator it = heap.begin();
    temp = it
```

[illegible]

```
        while(it != heap.end())
            if (it -> data < temp -> data)
                temp = it
            it++
          return temp;

list <Node*> extractmin (list<Node*> heap)

        list <Node*> newheap, lo;
        Node * temp;
        temp = getmin(heap)
       list<Node*> :: iterator it = heap.begin()
       while (it != heap.end())
          if (it != temp)
             newheap.push-back(it);
             it++
         lo = removeminFromtree ReturnBinHeap(temp);
         newheap = unionBinHeap(newheap, lo)
         newheap = adjust (newheap)
         return newheap;
```