IBM18 CS027
Dhairya K.

ADS

Binomial heap:-

Insertion in a binomial heap.

=>
```
list <Node *> insertATreeInHeap ( list <Node *> _heap, Node* 
                                   tree ).
    {
          list<Node *> temp;
           temp.push_back (tree);
            temp = union BiononialHeap (_heap, temp);
             return adjust(temp);

      }
```

=> removing minimum Key

```
list <Node *> removeMin FormTree returnHeap (Node * tree)
 {
      list <Node *> heap;
      Node *temp = tree->child;
      Node *lo;
      while (temp). {
               lo= temp;
               temp= temp->sibling ;
                lo -> sibling =NULL;
                heap . push_front(lo);
        }
        return heap;
   }.
```

1BM18CS027
Dharaj K.

```
//return min value node
Node * getMin (list <Node *> _heap)
{
        list <Node *>:: iterator it = _heap.begin();
        Node * temp = *it ;
        while (it != _heap.end()) {
                if ((*it)->data < temp->data)
                        temp = *it;
                it ++;
        }
        return temp;
}
```

```
//rearranging the heap.
list <Node *> adjust (list <Node *> _heap).
{
    if (_heap.size () <=1 )
            return _heap;
    list <Node*> new_heap;
    list <Node *> :: iterator it1, it2, it3;
    it1 = it2 = it3 = _heap.begin ();
    if (_heap.size ()== 2) {
                it2 = it1;
                it2++;
                it3 = _heap.end();
    } else {
            it2++;
            it3= it2;
            it3++;
    }
    while (it1 != _heap.end())
    {    if ( it2== _heap.end())
```

```
        it1++ ;
    else if ((*it1) -> degree < (*it2) -> degree).
    {       it1++;
            it2 ++;
             if (it3 != heap.end()).
                 it 3++;
            }

       else if ( (*it1)-> degree == (*it2) -> degree)
        {      node *temp;
               *it1 = merge BinomialTrees.(*it1, *it2);
               it2 = heap.erase(it2);
                if  (it3 != heap.end()).
                      it3++;
                }
             }
            return heap ;
    }
```