04/12/2020
Wednesday

ADS
Lab9

Kshitij R
1BM18CS045

classmate
Date
Page

```
struct Node{
        int data, degree;
        node * child, * sibling, * parent };
Node *  newNode(int data)
{       Node * t = new Node;
        t -> data = data
        t -> degree = 0
        t -> child = t -> parent = t -> sibling = NULL;
        return t;
    }
list <Node *> insertion of tree(list <Node *> heap,
                                Node * tree)
{   list <Node *> temp;
    temp.push_back(tree)
    temp = union of heap(heap, temp)
    return adjust(temp);
    }
list <Node *> unionofheap(list<Node*> l1, list
                                <Node*> l2)
{
    list<Node *> new;
    list <Node *> :: iterator it = l1.begin();
    list <Node *> :: iterator ot = l2.begin();
while(it != l1.end() && ot != l2.end())
{   if((it) -> degree <= (ot) -> degree)
        { new.push_back(it);
          it ++;
        }
```

Kshitij R
1BM18CS048

```
else
 { new.push_back(ot);
    ot++;
   }
}
 while(it != l1.end())
 {   new.push_back(*it);
     it++;
 }
while(ot != l2.end())
 {  new.push_back(*ot);
    ot++;
 }
 return  new;
}
list<Node*>insert( list<Node*> head, int data)
{ node * temp= newNode(data);
  return  insertion(head, temp)
 }
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; }
```

Kshitij.R
1BM18CS048

classmate
Date
Page

```
list <Node*> extractMin(list<Node *>heap)
{   list<Node*> newheap, lo;
    node * temp;
    temp = getMin(heap);
    list<Node*>:: iterator it;
    it= heap.begin();
    while (it != heap.end())
    {   if (*it != temp)
            { newheap.push_back(*it);
            }
        it ++;
    }
    lo=removeminandrettemp(temp);
    newheap= unionofheap(newheap, lo);
    newheap= adjust(newheap)
    return newheap;
    }
list<Node*>removeminandrettemp(Node *tree)
{ list <Node*>heap;
  Node* temp= tree->child;
   Node* lo;
   while(temp)
   { lo= temp;
      temp= temp->sibling;
      lo->sibling= NULL;
      heap.pushfront(lo)
      }
   return heap;
}
```