classmate
ARPANA M RAMASWAMY
1BM18CS147
09/12/2020
Arpana

PROGRAM -09

Q) WAP to implement the functions insert(H,K), getMin(H) and extractMin(H) on Binomial heap

```
struct Node
{  int val, deg;
   Node* parent, *child, *sibling;
};

Node* CreateNode (int n)
{
  Node * newN = new Node;
  newN->val=n;
   newN -> parent=NULL;
  newN -> sibling=NULL;
  newN -> child = NULL;
   newN -> deg = 0;
   return newN;
 }

Node* merge(Node*h1, Node*h2)
{
   if (h1==NULL) return h2;
  if( h2== NULL) return h1;
```

classmate
ARPANA M RAMACHANDRA
Page
1BM18CS147
09/12/2020
Biphany

```
Node *mergedNode = NULL;

if (h1->deg <= h2->deg)
    mergedNode = h1;

else if (h2->deg < h1->deg)
    mergedNode = h2;

while (h1 != NULL && h2 != NULL) {
    if (h1->deg < h2->deg) h1 = h1->sibling;
    else if (h1->deg == h2->deg)
    {
        Node *s = h1->sibling;
        h1->sibling = h2;
        h1 = s; }
    else {
        Node *s = h2->sibling;
        h2->sibling = h1;
        h2 = s; }
    }
    return mergedNode; }
}

Node* union(Node *h1, Node *h2)
{ if (h1 == NULL && h2 == NULL) return NULL;
```

Akshara M R
1BM18CS147
9/12/2020
Arjuna

```
Node *resNode = merge(h1, h2);
Node * prev = NULL, *curr = resNode, *next = curr->sibling;
while (next != NULL)
{
    if ((curr->deg != next->deg) || ((next->sibling
        != NULL) && (next->sibling)->deg ==
        curr->deg))
    {
        prev = curr;
        curr = next;
    }
    else
    {   ~~if (prev == NULL) resNode~~
if (curr->val <= next->val) {
        curr->sibling = next->sibling;
        binomialLink(next, curr);
    }
    else { if (prev == NULL) res = next;
        else prev->sibling = next;
        binomialLink(curr, next);
        curr = next; } }
    next = curr->sibling; }
    return resNode;
}
```

Afrana N R
1BM18CS147
@rfana

```
void Insert(Node* root, int x)
{ root = union(root, createNode(x)); }


Node* ExtractMin(Node *h)
{
  if (h==NULL) return NULL;

  Node *min-node_prev=NULL;
  Node *min-node=h;

  int min=h->val;
  Node* curr=h;
  while(curr->sibling!=NULL)
  {
    if((curr->sibling)->val<min)
    {
      min=(curr->sibling)->val;
      min-node-prev=curr;
      min-node=curr->sibling;
    }
    curr=curr->sibling; }

If(min-node-prev==NULL && min-node->sibling
   ==NULL) h=NULL;
```

ARPANA MR classmate
1BM18CS047
9/12/2020
Arpana

```
else if (min_node->prev == NULL)
        h = min_node->sibling;
else
        min_node->prev->sibling = min_node->sibling;

if (min_node->child != NULL)
{
    reverselist(min_node->child);
    (min_node->child)->sibling = NULL; }
    return union(h, root); }
}

Node* find (Node *h, int val)
{
    if (h == NULL) return NULL;
    if (h->val == val) return h;
    Node *resNode = findNode(h->child, val);
    if (resNode != NULL) return resNode;
    return findNode(h->sibling, val);
}

void decreaseKey (Node *H, int old_val, int new_v
{
    Node *node = find(H, old_val);
    if (node == NULL) return;
```

Arpana M R
1BM18CS147
@Arpana

```
node->val = new_val;
Node * parent = node->parent;

while (parent != NULL && node->val < parent->val)
{ swap(node->val, parent->val);
  node = parent;
  parent = parent->parent; }

Node * delete (Node *h, int val)
{
  if (h == NULL) return NULL;
  decreaseKey(h, val, minimum);
  return extractMin(h);
}
```

Arpana M R
1BM18CS147
@Arpana

```
Node* getMin(Node *h)
{
  if(h==NULL) return NULL;

  Node *min_node_prev=NULL;
  Node *min_node=h;
  int min = h->val
  Node* curr=h;
  while(curr->sibling!=NULL)
  {
    if((curr->sibling)->val<min)
    {
      min=(curr->sibling)->val;
      min_node_prev=curr;
      min_node=curr->sibling;
    }
    curr=curr->sibling;
  }
  return min_node;
}
```