MD YASEEN AHMED
1BM19CS404.

01. Write a program to implement the following functions on a Binomial heap.

(i) insert (H, K)
(ii) getMin (H)
(iii) extractMin (H)

```
// Structure definition for a Binomial
// Tree node

Struct node
{
    int data;
    Node *child, *sibling, *parent;
    int degree;
};

Node *newNode (int key)
{
```

```
    Node *temp = new Node;
    temp -> data = key;
    temp -> degree = 0;
    temp -> child = temp -> parent = temp ->
     siblings = NULL ;
    return temp;
}

list<Node*> insert (list<Node*> _head,
                    int key)
{
    Node *temp = newNode(key);
    return insertTreeHeap (_head, temp);
}

Node* getMin (list<Node*> _heap)
{
    list<Node*> :: iterator it = _heap.begin();
    Node *temp = *it;
    while (it != _head.end())
    {
      if ((*it)->data < temp->data)
          temp = *it;
```

```
        it++;
    }
    return temp;
}

list<Node*> extractMin(list<Node*> _heap)
{
    list<Node*> new_heap, lo;
    Node *temp;

    temp = getMin(_heap);
    list<Node*>:: iterator it;
    it = _heap.begin();

    while(it != _heap.end())
    {
        if (*it != temp)
            new_heap.push_back(*it);
        it++;
    }
    lo = removeMinFromTreeReturnBHeap(temp);
    new_heap = unionBinomialHeap(new_heap,
                    lo);
```

```
    new_heap = adjust(new_heap);
    return new_heap;
}

void printTree(Node *h)
{
    while(h)
    {
        cout << h->data << " ";
        printTree(h->child);
        h = h->sibling;
    }
}

void printHeap(list<Node*> _heap)
{
    list<Node*>::iterator it;
    it = _heap.begin();
    while(it != _heap.end())
    {
        printTree(*it);
        it++;
    }
}
```