Program - 4

Implementing the following functions on Binomial heap :-

1) Insert (H, k) :

```
insert ( list < Node * > _head , int key ) {
  Node * ~~temp~~ tree = newNode (key)
  list < Node * > temp;
  temp . push_back (tree);
  temp = Union Heap (_head, temp);
  return adjust (temp);

}.
UnionHeap ( 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) -> deg <= (*ot) -> deg) {
      _new . push_back (*it);
      it++;
   }
   else { _new . push_back (*ot);
          ot++;
   }
} while (it ! = l1 . end())
  { _new . push_back (*it);
      it++;
  } return new;
}
```

Shikha N
1BM18CS149
Shikha n

Shikha.N
1BM18CS149

(2) get Min (H) :

```
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;

}
```

(3) Extract_Min (H) :

```
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 = removemin (temp);
new_heap = UnionHeap(new_heap, lo);
new_heap = adjust (new_heap);
return new_heap;

}.
```

```
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 = it1 +1
              it3 = it2
               it3 ++;
    }
    while (it1 != _ heap.end()) {
    while (it1 != _heap.end()) it1 ++;
    if(it2 == _heap
    if(it2 ==_heap.end()) it1++;
    else if ((*it1)-> deg < (*it2)-> deg)
    { it1 ++;
        it2++;
      if (it3 != _ heap.end())
               it3 ++;
    }
    else if ((*it1)-> deg == (*it2)-> deg)
    { Node * temp;
        *it1 = mergeTree (*it1,*it2)
        it2 = _heap. erase (it2);
      if (it3 != _ heap. end())
                it3 ++;


      }
} return _heap;
}.
```

Shikha.N
1BM18CS149

[illegible]