Binomial Heap

Krishna prasad V
1BM19CS047
SA

```
fun Insert (head, ky):
{   Node * temp = new Node(ky)
    list <Node*> t
    t.push_back(temp)
    t = union BH (head, ky)
    return adjust (t)
}
adjust ( list <Node*> heap)
{   if (heap.size <=1) return heap
    list<Node*>  newheap
    auto 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()) it3++
}
else if (it3 != heap.end()) && (*it1 -> degree == *it2 -> degree && *it -> degree == *it3 -> degree)
    *it1 ++, *it2 ++, it3 ++;
}
return heap
}

func getmin ( list <Node*> heap)
{ auto it = heap.begin()
  while (it != heap.end())
  { if (*it -> data < temp -> data) temp = *it
    it ++
  }
  return temp
}

func extractmin( list< Node*> heap
{ list <Node*> newheap, lo, Nodeke hp;
  while (it != heap.end())
  { if (*it != temp) newheap.pushback(*it)
    it ++
  }
  lo = len (temp)
  newheap = union BH (newheap, lo)
  newheap = adjust (newheap)
  return newheap
}
```