4. 9

Binomial heap

Insert (heap, key);

{ Node temp = new node (key)

list < Node * > f

f . push - back (temp)

f = union (heap, key)

return adjust (f)

}

adjust (list < Node* > head)

{

if (heap.size <= 1) return heap

list < node * > new heap

auto i+1, i+2, i+3;

i+1 = i+2 = i+3 = heap. begin ();

if (heap - size () == 2)

{ i+2 = i+1

i+2++

i+3 = heap.end ()

}

else

{ i+2++

i+3 = i+2

i+3++

}

while (i+1 != heap end ())

```
{ if (i+2) == heapsize+1
  i++
  else if (i+1->degree < i+2->degree)
  { i++
    i+2++
    if (i+3 != heapsize)
    i+3++
  }
  else if(i+3 != heapsize && (i+1->deg == i+2->deg)
      && (i+1->deg == i+3->deg)
  { i+1++
    i+2++
    i+3++
  }
  return heap
}
```