ADS - 10

K. ARUN TEJA
1BM18CS041

(*) Write - Up

```
class BinomialHeap {   BinomialHeapNode Nodes;
                void
public ~~Reset~~ insert (int value) {
            if (value > 0) {
                    BinomialHeapNode temp = new BinomialHeapNode (value);
                    if( Nodes == null) {
                            Nodes = temp;
                            Size = 1;
                    }
                    else {
                            UnionNodes( temp);
                            Size++;
                    }.
            }
}.

void  merge ( BinomialHeapNode b) {
        BinomialHeapNode   temp1 = Nodes , temp2 = b;
        while (temp1 != null && temp2 != null) {
                if ( temp1.degree == temp2. degree) {
                        BinomialHeapNode tmp = temp2;
                        temp2 = temp2.sibling;
                        tmp.sibling = temp1. sibling;
                        temp1. sibling = tmp;
                        temp1 = tmp.sibling;
                }
                else {
                    if ( temp1. degree < temp2.degee) {
                        if ( temp1. sibling == null || temp1. sibling. degree >
                                                                temp2. degee) {
                                BinomialHeapNode tmp = temp2;
                                temp2 = temp2. sibling;
                                tmp. sibling = temp1. sibling;
                                ~~temp~~1.sibling = tmp;
                                temp1 = tmp sibling ;
                        }
                        else {
                                temp1 = temp1.sibling;
                        }
                    }.
                    else {
                        BinomialHeapNode tmp = temp1;
                        temp1 = temp2;
                        temp2 = temp2. Sibling;
                        temp1. Sibling = tmp;
                        if ( tmp == Nodes) {
                                Nodes = temp1;
                        }
                    }.
                }.
        }.
        if (temp1 ==null) {
                temp1= Nodes;
                while( temp1. sibling != null) {
                                temp1 = temp1. sibling;
                }
                temp1. Sibling = temp2;
        }
    }.

void    unionNodes ( BinomialHeapNode  b) {

        merge (b) ;
        BinomialHeapNode   prevTemp = null,  temp = Nodes ,  next Temp = Nodes.sibling;
        while (next Temp != null) {
                if( temp.degree != next Temp. degree || nextTemp . sibling != null &&
                        nextTemp. sibling. degree == temp. degree) {

                        prevTemp= temp;
                        temp = nextTemp;
                }.
                else {
                        if ( temp. key <= nextTemp .key) {
                                temp. sibling = nextTemp. sibling;
                                nextTemp . parent = temp;
                                next Temp. Sibling = temp child;
                                temp. child = next Temp;
                                temp.degree ++;
                        }.
                        else {
                                if (prevTemp == null) {
                                        Nodes = next Temp;
                                }
                                else {
                                        prevTemp. sibling = next Temp;
                                }.
                                temp parent = nextTemp;
                                temp. sibling = next Temp.child ;
                                next Temp. child = temp ;
                                next Temp. degree ++;
                                temp= nextTemp;
                        }
                }.
                nextTemp = temp. sibling;
        }
}
```

```
public int findMin () {
        BinomialHeapNode x = Nodes.key, y ;
        int  min = x.key ;
        while ( x != null ) {
                if( x.key < min ) {
                        y = x ;
                        min = x.key;
                }
                x = x.sibling;
        }
        return y;
}

public int ExtractMin () {
        if ( Nodes == null )
                return -1;
        BinomialHeapNode  temp = Nodes, prevTemp = null;
        BinomialHeapNode  minNode = Nodes.findMin ();
        while (temp.key != minNode.key) {
                prevTemp = temp;
                temp = temp.sibling;
        }
        if ( prevTemp == null) {
                Nodes = temp.sibling;
        }
        else {
                prevTemp.sibling = temp.sibling;
        }
        temp = temp.child;
        BinomialHeapNode f = temp;
        while (temp != null) {
                temp.parent = null;
                temp = temp.sibling;
        }
        if (Nodes == null && f == null) {
                Size = 0;
        }
        else {
                if ( Nodes == null && f != null) {
                        Nodes = f.reverse ();
                        Size = Nodes.getSize ();
                }
                else {
                        if ( Nodes != null && f == null ) {
                                Size = Nodes.getSize ();
                        }
                        else {
                                unionNodes (f.reverse ());
                                Size = Nodes.getSize ();
                        }
                }
        }
        return minNode.key;
}
```