Name : Rajkumar B L
Reg.No : 2047120
Course : MCS 271 Data Structure(Lab 15-Binomial Heap Tree)

Code:-

```c
/*********************
 * Name : Rajkumar B L
 * Reg  : 2047120
 * Lab  : 15
 * Program : Binomial Heap
 * ******************/
#include <stdio.h>
#include <malloc.h>

struct node
{
    int n;
    int degree;
    struct node *parent;
    struct node *child;
    struct node *sibling;
};

struct node *MAKE_bin_HEAP();
int bin_LINK(struct node *, struct node *);
struct node *CREATE_NODE(int);
struct node *bin_HEAP_UNION(struct node *, struct node *);
struct node *bin_HEAP_INSERT(struct node *, struct node *);
struct node *bin_HEAP_MERGE(struct node *, struct node *);
struct node *bin_HEAP_EXTRACT_MIN(struct node *);
int REVERT_LIST(struct node *);
int DISPLAY(struct node *);
struct node *FIND_NODE(struct node *, int);
int bin_HEAP_DECREASE_KEY(struct node *, int, int);
int bin_HEAP_DELETE(struct node *, int);

int count = 1;

struct node *MAKE_bin_HEAP()
{
    struct node *np;
```

```
    np = NULL;
    return np;
}

struct node *H = NULL;
struct node *Hr = NULL;

int bin_LINK(struct node *y, struct node *z)
{
    y->parent = z;
    y->sibling = z->child;
    z->child = y;
    z->degree = z->degree + 1;
}

struct node *CREATE_NODE(int k)
{
    struct node *p; //new node;
    p = (struct node *)malloc(sizeof(struct node));
    p->n = k;
    return p;
}

struct node *bin_HEAP_UNION(struct node *H1, struct node *H2)
{
    struct node *prev_x;
    struct node *next_x;
    struct node *x;
    struct node *H = MAKE_bin_HEAP();
    H = bin_HEAP_MERGE(H1, H2);
    if (H == NULL)
        return H;
    prev_x = NULL;
    x = H;
    next_x = x->sibling;
    while (next_x != NULL)
    {
        if ((x->degree != next_x->degree) || ((next_x->sibling != NULL) && (next_x->sibling)-
>degree == x->degree))
        {
            prev_x = x;
            x = next_x;
        }
        else
        {
            if (x->n <= next_x->n)
            {
                x->sibling = next_x->sibling;
                bin_LINK(next_x, x);
            }
            else
```

```
            {
                if (prev_x == NULL)
                    H = next_x;
                else
                    prev_x->sibling = next_x;
                bin_LINK(x, next_x);
                x = next_x;
            }
        }
        next_x = x->sibling;
    }
    return H;
}

struct node *bin_HEAP_INSERT(struct node *H, struct node *x)
{
    struct node *H1 = MAKE_bin_HEAP();
    x->parent = NULL;
    x->child = NULL;
    x->sibling = NULL;
    x->degree = 0;
    H1 = x;
    H = bin_HEAP_UNION(H, H1);
    return H;
}

struct node *bin_HEAP_MERGE(struct node *H1, struct node *H2)
{
    struct node *H = MAKE_bin_HEAP();
    struct node *y;
    struct node *z;
    struct node *a;
    struct node *b;
    y = H1;
    z = H2;
    if (y != NULL)
    {
        if (z != NULL && y->degree <= z->degree)
            H = y;
        else if (z != NULL && y->degree > z->degree)
            /* need some modifications here;the first and the else conditions can be merged together
!!!! */
            H = z;
        else
            H = y;
    }
    else
        H = z;
    while (y != NULL && z != NULL)
    {
        if (y->degree < z->degree)
```

```
        {
            y = y->sibling;
        }
        else if (y->degree == z->degree)
        {
            a = y->sibling;
            y->sibling = z;
            y = a;
        }
        else
        {
            b = z->sibling;
            z->sibling = y;
            z = b;
        }
    }
    return H;
}

int DISPLAY(struct node *H)
{
    struct node *p;
    if (H == NULL)
    {
        printf("\nHeap empty");
        return 0;
    }
    printf("\nThe root nodes are :-\n");
    p = H;
    while (p != NULL)
    {
        printf("%d", p->n);
        if (p->sibling != NULL)
            printf("-->");
        p = p->sibling;
    }
    printf("\n");
}

struct node *bin_HEAP_EXTRACT_MIN(struct node *H1)
{
    int min;
    struct node *t = NULL;
    struct node *x = H1;
    struct node *Hr;
    struct node *p;
    Hr = NULL;
    if (x == NULL)
    {
        printf("\nNothing to extract");
        return x;
```

```
    }
    //    int min=x->n;
    p = x;
    while (p->sibling != NULL)
    {
        if ((p->sibling)->n < min)
        {
            min = (p->sibling)->n;
            t = p;
            x = p->sibling;
        }
        p = p->sibling;
    }
    if (t == NULL && x->sibling == NULL)
        H1 = NULL;
    else if (t == NULL)
        H1 = x->sibling;
    else if (t->sibling == NULL)
        t = NULL;
    else
        t->sibling = x->sibling;
    if (x->child != NULL)
    {
        REVERT_LIST(x->child);
        (x->child)->sibling = NULL;
    }
    H = bin_HEAP_UNION(H1, Hr);
    return x;
}

int REVERT_LIST(struct node *y)
{
    if (y->sibling != NULL)
    {
        REVERT_LIST(y->sibling);
        (y->sibling)->sibling = y;
    }
    else
    {
        Hr = y;
    }
}

struct node *FIND_NODE(struct node *H, int k)
{
    struct node *x = H;
    struct node *p = NULL;
    if (x->n == k)
    {
        p = x;
        return p;
```

```
    }
    if (x->child != NULL && p == NULL)
    {
        p = FIND_NODE(x->child, k);
    }

    if (x->sibling != NULL && p == NULL)
    {
        p = FIND_NODE(x->sibling, k);
    }
    return p;
}

int bin_HEAP_DECREASE_KEY(struct node *H, int i, int k)
{
    int temp;
    struct node *p;
    struct node *y;
    struct node *z;
    p = FIND_NODE(H, i);
    if (p == NULL)
    {
        printf("\nInvalid choice of key to be reduced");
        return 0;
    }
    if (k > p->n)
    {
        printf("\nSorry! the new key is greater than current one");
        return 0;
    }
    p->n = k;
    y = p;
    z = p->parent;
    while (z != NULL && y->n < z->n)
    {
        temp = y->n;
        y->n = z->n;
        z->n = temp;
        y = z;
        z = z->parent;
    }
    printf("\nKey reduced successfully!");
}

int bin_HEAP_DELETE(struct node *H, int k)
{
    struct node *np;
    if (H == NULL)
    {
        printf("\nHeap empty!");
        return 0;
```

```
    }

    bin_HEAP_DECREASE_KEY(H, k, -1000);
    np = bin_HEAP_EXTRACT_MIN(H);
    if (np != NULL)
        printf("\nNode deleted successfully!");
}

int main()
{

    printf("\n***************************\n*  Name : Rajkumar B L     *\n*  Reg  : 2047120
 *\n*  Lab  : 15               *\n*  Prg  : Binomial Heap    *\n***************************\n\n");

    int i, n, m, l;
    struct node *p;
    struct node *np;
    char ch;
    printf("\nEnter the number of elements : ");
    scanf("%d", &n);
    printf("\nEnter the %d elements : ",n);
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &m);
        np = CREATE_NODE(m);
        H = bin_HEAP_INSERT(H, np);
    }
    DISPLAY(H);
    do
    {
        printf("\n===========================\n\t   Menu\n===========================\n");
        printf("1.Insert an element\n2.Extract the minimum key\n3.Decrease a node key\n4.Delete a no
de\n5.Quit\n");
        printf("===========================\n");
        printf("Enter your choice: ");
        scanf("%d", &l);
        switch (l)
        {
        case 1:
            do
            {
                printf("\nEnter the element to be inserted : ");
                scanf("%d", &m);
                p = CREATE_NODE(m);
                H = bin_HEAP_INSERT(H, p);
                //printf("\nNow the heap is:\n");
                DISPLAY(H);
                printf("\nInsert More(y/Y) : ");
                //fflush(stdin);
                scanf("%s", &ch);
            } while (ch == 'Y' || ch == 'y');
```

```
            break;
        case 2:
            do
            {
                printf("\nExtracting the minimum key node");
                p = bin_HEAP_EXTRACT_MIN(H);
                if (p != NULL)
                    printf("\nThe extracted node is %d", p->n);
                //printf("\nNow the heap is :\n");
                DISPLAY(H);
                printf("\nExtract More(y/Y) : ");
                fflush(stdin);
                scanf("%s", &ch);
            } while (ch == 'Y' || ch == 'y');
            break;
        case 3:
            do
            {
                printf("\nEnter the key of the node to be decreased : ");
                scanf("%d", &m);
                printf("\nEnter the new key : ");
                scanf("%d", &l);
                bin_HEAP_DECREASE_KEY(H, m, l);
                //printf("\nNow the heap is:\n");
                DISPLAY(H);
                printf("\nDecrease More(y/Y) : ");
                fflush(stdin);
                scanf("%s", &ch);
            } while (ch == 'Y' || ch == 'y');
            break;
        case 4:
            do
            {
                printf("\nEnter the key to be deleted : ");
                scanf("%d", &m);
                bin_HEAP_DELETE(H, m);
                printf("\nDelete More(y/Y) : ");
                fflush(stdin);
                scanf("%s", &ch);
            } while (ch == 'y' || ch == 'Y');
            break;
        case 5:
            printf("Bye!\n\n");
            break;
        default:
            printf("\nInvalid Choice, Try Again!\n");
        }
    } while (l != 5);
}
```

Output:-

Ubuntu 20.04 LTS

```
kumarraj@kumarraj:~/MCS_271/Labs/Lab15$ gcc lab15.c
kumarraj@kumarraj:~/MCS_271/Labs/Lab15$ ./a.out

****************************
*  Name : Rajkumar B L     *
*  Reg  : 2047120          *
*  Lab  : 15               *
*  Prg  : Binomial Heap    *
****************************


Enter the number of elements : 5

Enter the 5 elements : 13 25 36 47 58

The root nodes are :-
58-->13

============================
           Menu
============================
1.Insert an element
2.Extract the minimum key
3.Decrease a node key
4.Delete a node
5.Quit
============================
Enter your choice: 1

Enter the element to be inserted : 69

The root nodes are :-
58-->13

Insert More(y/Y) : y

Enter the element to be inserted : 72

The root nodes are :-
72-->58-->13
```

Ubuntu 20.04 LTS

```
============================
           Menu
============================
1.Insert an element
2.Extract the minimum key
3.Decrease a node key
4.Delete a node
5.Quit
============================
Enter your choice: 3

Enter the key of the node to be decreased : 58

Enter the new key : 56

Key reduced successfully!
The root nodes are :-
72-->56-->13

Decrease More(y/Y) : y

Enter the key of the node to be decreased : 72

Enter the new key : 70

Key reduced successfully!
The root nodes are :-
70-->56-->13

Decrease More(y/Y) : n
```

```
============================
           Menu
============================
1.Insert an element
2.Extract the minimum key
3.Decrease a node key
4.Delete a node
5.Quit
============================
Enter your choice: 2

Extracting the minimum key node
The extracted node is 13
The root nodes are :-
72-->58

Extract More(y/Y) : y

Extracting the minimum key node
The extracted node is 58
The root nodes are :-
72

Extract More(y/Y) : y

Extracting the minimum key node
The extracted node is 72
Heap empty
Extract More(y/Y) : y

Extracting the minimum key node
Nothing to extract
Heap empty
Extract More(y/Y) : n
```

```
============================
           Menu
============================
1.Insert an element
2.Extract the minimum key
3.Decrease a node key
4.Delete a node
5.Quit
============================
Enter your choice: 4

Enter the key to be deleted : 13

Key reduced successfully!
Node deleted successfully!
Delete More(y/Y) : n

============================
           Menu
============================
1.Insert an element
2.Extract the minimum key
3.Decrease a node key
4.Delete a node
5.Quit
============================
Enter your choice: 5
Bye!
```