Page No.
Date

Program-7

(9) Write a program to implement insertion operation on a B-tree.

```
Struct BTree {
    int *d;
    BTree **child_ptr;
    bool l;
    int n;
} *r = NULL, *np = NULL, *x = NULL;
int n;
BTree* init() {
    int i;
    np = new BTree;
    np->d = new int[n];
    np->child_ptr = new BTree *[n+1];
    np->l = true;
    np->n = 0;
    for (i=0; i<n+1; i++) {
        np->child_ptr[i] = NULL;
    }
    return np;
}

void Sort (int *p, int n) {
    int i, j, t;
    for(i=0; i<n; i++) {
        for (j=1; j<=n; j++) {
            if (p[i] > p[j]) {
                swap(p[i], p[j])
            }
        }
    }
}
```

```
int split_child (BTree *x, int i) {
    int j, mid;
    BTree *np1, *np3, *y;
    np3 = init ();
    np3->l = true;
    if (r == -1) {
        mid = x->d[n/2];
        x->d[n/2] = 0;

        x->n--;
        np1 = init();
        np1->l = false;
        x->l = false; x->l = true;
        for (j=n/2+1; j<n; j++) {

            np3->d[j-n/2-1] = x->d[j];

            np3->child_ptr[j-n/2-1] =
                                x->child_ptr[j];

            np3->n++;

            x->d[j]=0;

            x->n--;

        for(j=0; j<n; j++) {

            x->child_ptr[j] = NULL;
        }
        np1->d[0]=mid;
        np1->child_ptr[np1->n] = x
        np1->child_ptr[np1->n+1] = np3
        np1->n++;
        r = np1;
}
```

(E) Jumanth.V

```
else {
    y = x->child_ptr[i];
    mid = y->d[n/2];

    y->d[2] = 0;
    y->n--;
    for(j = n/2+1; j<n; j++) {

        np3->d[j-n/2-1] = y->d[j];
        np3->n++;
        y->d[j] = 0;
        y->n--;
    }
    x->child_ptr[i+1] = y;
    x->child_ptr[i+1] = np3;
}
return mid;
}

void insert (int a) {
    int i, t;
    x = r;
    if (x == NULL) {
        r = init();
        x = r;

    }
    else {
    if (x->l == true && x->n == n) {
        t = split_child(x, -1);
        x = r;
        for(i=0; i<(x->n); i++) {
            if ((a > x->d[i]) &&
                (a < x->d[i+1])) {
```

```
            i++;
            break;
        }
    else if (a<x->d[0]) {
            break;
        }
    else {
        continue;
        }
    }
    x = x->child_ptr[i];
}
else {
    while (x->l == false) {
        for(i=0; i<(x->n); i++) {
        if ((a>x->d[i]) &&
            (a<x->d[i+1])) {
            i++;
            break;
        }
    else if (a<x->d[0]) {
            break;
    }
    else {
        continue;
    }
    }
    if ((x->child_ptr[i])->n == m) {
        t = split_child(x,i);
        x->d[x->n] = t;
        x->n++;
        continue;
    }
    else {
```

```
            x = x->child_ptr[i];
        }
    }
}
}
x->d[x->n] = a;
sort(x->d, x->n);
x->n++;
}
```

(5) Jumanth.K