LEMBAR KERJA PRAKTIKUM 14
STRUKTUR DATA HIRARKI BAGIAN III (B TREE)

Soal

1. Pada potongan kode (snippet) nomor 2, jelaskan hasil eksekusi dari baris kode tersebut?

```
struct node* searchforleaf(struct node* root, int k, struct node*
parent, int chindex) {
    if (root) {
        if (root->isleaf == 1)
            return root;
        else {
            int i;
            if (k < root->key[0])
                root = searchforleaf(root->child[0], k, root, 0);
            else {
                for (i = 0; i < root->n; i++)
                    if (root->key[i] > k)
                        return searchforleaf(root->child[i], k,
root, i);
                return searchforleaf(root->child[i], k, root, i);
            }
        }
    } else {
        struct node* newleaf = new struct node;
        newleaf->isleaf = 1;
        newleaf->n = 0;
        parent->child[chindex] = newleaf;
        newleaf->parent = parent;
        return newleaf;
    }
}
```

Potongan kode ini digunakan untuk mencari node leaf yang sesuai untuk insertion elemen baru yang kemudian return node leaf tersebut. Lalu terdapat baris if (root->isleaf == 1) yang memeriksa apakah posisi node adalah daun/leaf atau tidak. Jika benar merupakan leaf maka return node tersebut untuk insertion. Jika bukan leaf maka kode akan memilih node child berdasarkan nilai k dibandingkan key[] yang kemudian direkursifkan dengan fungsi searchforleaf(,,,) pada node child tersebut. Apabila root null mka dibuat leaf baru

pada posisi chindex dan lalu direturn-kan nodenya untuk menjadi tempat insertion leaf selanjutnya.

2. Barikan gambaran hasil eksekusi dari kode (snippet) nomor 3? Barikan ***highlight/block*** dari potongan baris dan kaitkan dengan penjelasan dari salah satu karakteristik algoritme B-Tree

```
if (p->n < N - 1) {
      int i;
      for (i = 0; i < p->n; i++) {
            if (p->key[i] > e) {
                  for (int j = p->n - 1; j >= i; j--)
                        p->key[j + 1] = p->key[j];
                        break;
                  }
            }
            p->key[i] = e;
            p->n = p->n + 1;
            return root;
}
```

Kode diatas menggambarkan karakteristik B Tree yang memiliki lebih dari 1 key. Jika node p (leaf) masih menyimpan kurang dari N-1 key, maka elemen e akan disisipkan di posisi yang menjaga urutan menaik. Key di kanan posisi itu digeser satu langkah ke kanan. Setelah sisipan, fungsi langsung kembali tanpa perlu split.

3. Lengkapi potongan kode (snippet) nomor 4, dan implementasikan keseluruhan kode agar dapat menjalankan perintah kode B-Tree dan perlihatkan hasilnya

```
int main() {
    /* Jika Tree beriut ini sudah dikonstruksi
    Dengan Root 6 dan memiliki anak kiri (1, 2, 4) dan kanan (7,8,9)
          6
        /   \
    1 2 4   7 8 9

    Kemudian ditambahkan elemen nilai 5, dan B-Tree menjadi:
        4   7
      /     \     \
    1 2   5 6   8 9
    */
    // Mulai dengan Empty Root
```

```
    struct node* root = NULL;

    // Tambahkan 6
    root = insert(root, 6);

    // Tambahkan 1, 2, 4 Ke kiri dari 6
    root->child[0] = insert(root->child[0], 1);
    root->child[0] = insert(root->child[0], 2);
    root->child[0] = insert(root->child[0], 4);
    root->child[0]->parent = root;

    // Tambahkan 7, 8, 9 Ke kanan dari 6
    root->child[1] = insert(root->child[1], 7);
    root->child[1] = insert(root->child[1], 8);
    root->child[1] = insert(root->child[1], 9);
    root->child[1]->parent = root;

    // Cetak Tree Original
    cout << "Original tree: " << endl;
    for (int i = 0; i < root->n; i++)
        cout << "     " << root->key[i] << " ";
    cout << endl;
    for (int i = 0; i < 2; i++) {
        cout << root->child[i]->key[0] << " ";
        cout << root->child[i]->key[1] << " ";
        cout << root->child[i]->key[2] << " ";
        cout << " ";
    }
    cout << endl;

    // Tambahkan Nilai 5
    cout << "Setelah menambakan nilai 5: " << endl;
    root->child[0] = insert(root->child[0], 5);

    // Cetak Tree hasil penambahan dengan elemen 5
    cout << "     ";
    for (int i = 0; i <= root->n; i++)
        cout << root->key[i] << " ";
    cout << endl;
    for (int i = 0; i < N - 1; i++) {
        cout << root->child[i]->key[0] << " ";
        cout << root->child[i]->key[1] << "  ";
    }

    return 0;
}
```

Output Program:

```
Original tree:
     6
1 2 4  7 8 9
Setelah menambakan nilai 5:
     4 7
1 2  5 6  8 9
```

4. Tuliskan *driver code* ***(int main)*** dengan memodifikasi baris kode pada nomor 4 yang memperlihatkan bentuk B-Tree yang sudah dikonstruksi:
   Dengan Node root adalah 9, dan 24. Kemudian memiliki anak kiri bagi 9 (7, 6, 8), anak kanan bagi 9 atau anak kiri bagi 24 (12, 17 19), dan anak kanan bagi 24 (29, 31, 41) Cetak hasilnya dan perlihatkan hasil ***printscreen***

```
int main() {

    // Mulai dengan Empty Root
    struct node* root = NULL;

    // Tambahkan 9 dan 24
    root = insert(root, 9);
    root = insert(root, 24);

    // Tambahkan 7, 6, 8 Ke kiri dari 9
    root->child[0] = insert(root->child[0], 7);
    root->child[0] = insert(root->child[0], 6);
    root->child[0] = insert(root->child[0], 8);
    root->child[0]->parent = root;

    // Tambahkan 12, 17, 19 Ke kanan dari 9 dan kiri dari 24
    root->child[1] = insert(root->child[1], 12);
    root->child[1] = insert(root->child[1], 17);
    root->child[1] = insert(root->child[1], 19);
    root->child[1]->parent = root;

    // Tambahkan 29, 31, 41 Ke kanan dari 24
    root->child[2] = insert(root->child[2], 29);
    root->child[2] = insert(root->child[2], 31);
    root->child[2] = insert(root->child[2], 41);
    root->child[2]->parent = root;
```

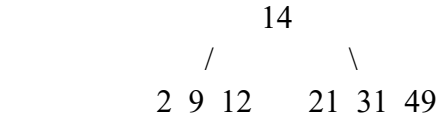
```
    // Cetak Tree Original
    cout << "Original tree: " << endl;
    for (int i = 0; i < root->n; i++)
        cout << "     " << root->key[i] << " ";
    cout << endl;
    for (int i = 0; i < N-1; i++) {
        cout << root->child[i]->key[0] << " ";
        cout << root->child[i]->key[1] << " ";
        cout << root->child[i]->key[2] << " ";
        cout << " ";
    }
    cout << endl;

    return 0;
}
```

Output Program

```
Original tree:
     9      24
6 7 8  12 17 19  29 31 41
```

5. Tuliskan *driver code* ***(int main)*** dengan memodifikasi baris kode pada nomor 4 yang memperlihatkan bentuk B-Tree yang sudah dikonstruksi:
   Dengan Node root adalah 14 dan memiliki anak kiri (12, 2, 9) dan kanan (49, 21, 31)



   Jika ditambahkan elemen nilai 17, cetak hasilnya dan perlihatkan hasil ***printscreen*** dari penambahan elemen tersebut.

```
int main() {
    // Mulai dengan Empty Root
    struct node* root = NULL;

    // Tambahkan 14
    root = insert(root, 14);

    // Tambahkan 12, 2, 9 Ke kiri dari 14
    root->child[0] = insert(root->child[0], 12);
    root->child[0] = insert(root->child[0], 2);
    root->child[0] = insert(root->child[0], 9);
    root->child[0]->parent = root;
```

```
    // Tambahkan 49, 21, 31 Ke kanan dari 14
    root->child[1] = insert(root->child[1], 49);
    root->child[1] = insert(root->child[1], 21);
    root->child[1] = insert(root->child[1], 31);
    root->child[1]->parent = root;

    // Cetak Tree Original
    cout << "Original tree: " << endl;
    for (int i = 0; i < root->n; i++)
        cout << "     " << root->key[i] << " ";
    cout << endl;
    for (int i = 0; i < 2; i++) {
        cout << root->child[i]->key[0] << " ";
        cout << root->child[i]->key[1] << " ";
        cout << root->child[i]->key[2] << " ";
        cout << " ";
    }
    cout << endl;

    cout << "Setelah insert nilai 17: " << endl;
    root->child[0] = insert(root->child[0], 17);

    cout << "     ";
    for (int i = 0; i <= root->n; i++)
        cout << root->key[i] << " ";
    cout << endl;
    for (int i = 0; i < N - 1; i++) {
        cout << root->child[i]->key[0] << " ";
        cout << root->child[i]->key[1] << "  ";
    }

    return 0;
}
```

Output Program:

```
Original tree:
     14
2 9 12  21 31 49
Setelah menambakan nilai 17:
     12 21
2 9  17 14  31 49
```