

**LAPORAN UJIAN AKHIR SEMESTER GENAP**  
**ALGORITMA DAN DATA STRUKTUR**

**PIZZAHUT (FOOD & DRINKS ORDERING )**



**UMN**  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

**Kevin Mikael - 00000111440**

**Bagus Kuncoro A Y - 00000113628**

**Christian Surya T - 00000116930**

**Adam Rifqy Hajat - 000001133876**

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS TEKNIK DAN INFORMATIKA**  
**UNIVERSITAS MULTIMEDIA NUSANTARA**

**2025**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>0</b>
<b>BAB 1 PENDAHULUAN.....</b>	<b>1</b>
1.1. Sumber.....	1
1.3. Batasan Masalah.....	2
1.4. Tujuan dan Manfaat.....	2
<b>BAB 2 PEMBAHASAN.....</b>	<b>3</b>
2.1. Soal 1 : Sub-CPMK0643.....	3
2.2. Soal 2 : Sub-CPMK0643.....	12
2.3. Soal 3 : Sub-CPMK0644.....	20
2.4. Soal 4 : Sub-CPMK0644.....	24
<b>BAB 3 PEMBAGIAN TUGAS.....</b>	<b>28</b>
3.1 Ketua Kelompok.....	28
3.2 Anggota Kelompok.....	28
<b>BAB 4 EVALUASI.....</b>	<b>29</b>
4.1 Ketua Kelompok.....	29
4.2 Anggota Kelompok.....	29
<b>BAB 5 KESIMPULAN.....</b>	<b>30</b>

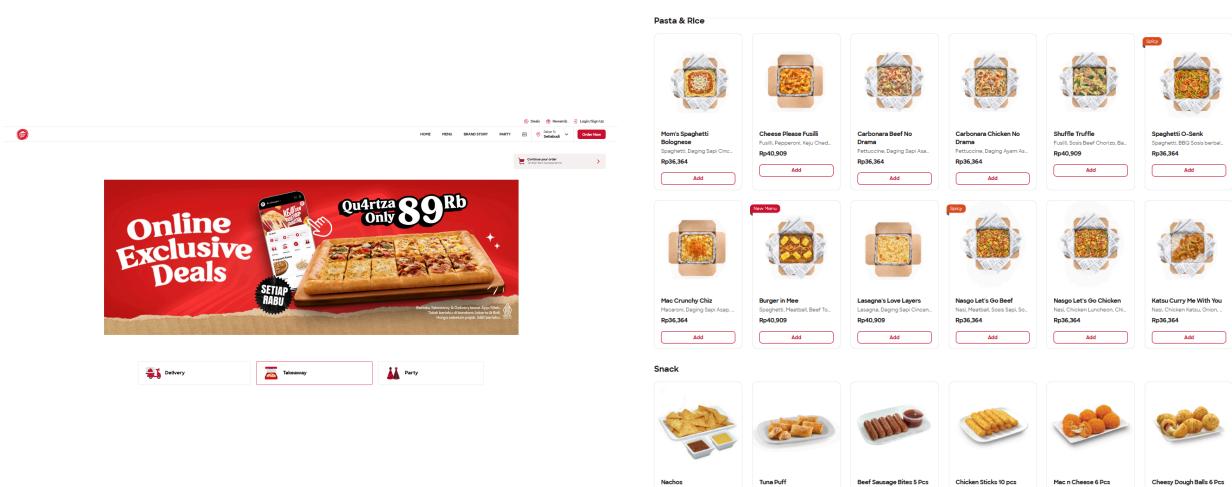
# BAB 1

## PENDAHULUAN

### 1.1. Sumber

Sebagai sumber dan referensi untuk merancang aplikasi PizzaHut (Food and Drinks Ordering), kami mengambil dari website milik Pizza Hut Indonesia (<https://www.pizzahut.co.id>). Dari website tersebut, ditemukan beberapa fitur utama dan alur proses bisnis yang menjadi dasar dalam perancangan program aplikasi pemesanan makanan. Proses bisnis dimulai dari pengguna yang mengakses daftar menu makanan dan minuman yang tersedia, seperti pizza, pasta, minuman, dan lainnya. Beberapa item dalam daftar tersebut disertai dengan pilihan tambahan seperti topping. Pengguna dapat memilih menu yang diinginkan serta menyesuaikan pesanan sesuai preferensi melalui opsi kustomisasi topping. Setelah selesai memilih, pengguna dapat menambahkan item ke dalam keranjang pesanan.

Pada tahap berikutnya, pengguna memiliki opsi untuk mengubah jumlah atau menghapus item yang telah dimasukkan ke dalam keranjang sebelum melanjutkan ke proses checkout. Proses checkout mencakup perhitungan total biaya berdasarkan harga menu, topping tambahan, dan jumlah pesanan. Selain itu, pengguna diminta untuk mengisi data diri seperti nama dan alamat pengiriman serta memilih metode pembayaran yang tersedia. Setelah pemesanan dikonfirmasi, sistem akan menyimpan riwayat pesanan dan menampilkan status pesanan secara berkala, mulai dari status "Menunggu", "Diproses", hingga "Selesai".



## **1.2. Rumusan Masalah**

- ❖ Bagaimana mencari, memasukkan, dan menghapus node pada Binary Search Tree?
- ❖ Bagaimana mencari, memasukkan, dan menghapus node pada Binary Heap?
- ❖ Bagaimana mengimplementasikan algoritma bubble dan insertion sort pada tampilan menu?
- ❖ Bagaimana mengimplementasikan algoritma binary search pada menu?

## **1.3. Batasan Masalah**

- ❖ Aplikasi ditulis menggunakan bahasa C dengan sistem operasi Windows.
- ❖ Program hanya menangani dua jenis menu, yaitu makanan dan minuman yang dibaca dari file teks.
- ❖ Tidak ada antarmuka grafis (GUI) maupun koneksi ke sistem online.
- ❖ Struktur data yang digunakan hanya Binary Search Tree dan Binary Max-Heap.
- ❖ Sorting hanya dilakukan dengan metode Bubble Sort dan Insertion Sort, tergantung pilihan pengguna.
- ❖ Pencarian menu hanya menggunakan Binary Search, dan hanya bisa dilakukan setelah data diurutkan.
- ❖ Harga menu dianggap integer.

## **1.4. Tujuan dan Manfaat**

### **a) Tujuan**

- ❖ Membangun aplikasi pemesanan makanan dan minuman berbasis teks yang interaktif dan efisien.
- ❖ Memudahkan pelanggan dalam memilih menu makanan dan minuman secara fleksibel.
- ❖ Menyediakan sistem keranjang belanja yang dapat diubah sebelum proses checkout.
- ❖ Membiasakan penggunaan file untuk input menu eksternal.
- ❖ Meningkatkan pemahaman mengenai BST, Max-Heap, Bubble Sort, Insertion Sort, dan Binary Search

### **b) Manfaat**

- ❖ Pengguna dapat memesan makanan dengan cepat tanpa harus datang langsung ke kasir.
- ❖ Fitur visual (seperti bentuk pizza) dan sistem menu dinamis memberikan pengalaman yang menyenangkan meskipun berbasis teks.
- ❖ Memberikan gambaran implementasi struktur data BST dan Max-Heap dalam aplikasi pemesanan menu makanan dan minuman.

## BAB 2

### PEMBAHASAN

#### 2.1. Soal 1 : Sub-CPMK0643

Binary Search Tree kami implementasikan pada jenis order **makan di tempat**.

```
typedef struct BSTNode {
    char name[100];
    int harga;
    struct BSTNode* left;
    struct BSTNode* right;
} BSTNode;
```

Berikut adalah **linked list** untuk Binary Search Tree (BST) sebagai struktur data.

```
BSTNode* insertBST(BSTNode* root, menuStruct data) {
    if (root == NULL) {
        BSTNode* newNode = (BSTNode*)malloc(sizeof(BSTNode));
        strcpy(newNode->name, data.name);
        newNode->harga = data.harga;
        newNode->left = newNode->right = NULL;
        return newNode;
    }

    if (data.harga < root->harga)
        root->left = insertBST(root->left, data);
    else
        root->right = insertBST(root->right, data);

    return root;
}
```

Berikut adalah, fungsi **insertBST** untuk menambahkan node ke dalam Binary Search Tree dengan harga sebagai value dari node.

```
BSTNode* tambahPesananbst(menuStruct makanan[], menuStruct minuman[], BSTNode* root) {
    int kategori, id;
    menuStruct pesanan;

    printf("Pilih kategori:\n");
    printf("[1] Makanan\n[2] Minuman\n");
    printf("Masukkan pilihan: ");
    scanf("%d", &kategori);
    if (kategori == 1) {
        printf("Masukkan ID makanan: ");
        scanf("%d", &id);

        if (id <= 0 || id > 40) {
            printf("ID tidak valid!\n");
            return root;
        }
        pesanan = makanan[id - 1];
    } else if (kategori == 2) {
        printf("Masukkan ID minuman: ");
        scanf("%d", &id);

        if (id <= 0 || id > 40) {
            printf("ID tidak valid!\n");
            return root;
        }
        pesanan = minuman[id - 1];
    } else {
        printf("Kategori tidak valid.\n");
        return root;
    }

    // Masukkan ke dalam keranjang BST
    root = insertBST(root, pesanan);
    printf("Pesanan berhasil ditambahkan ke keranjang!\n");
    return root;
}
```

Berikut adalah fungsi **tambahPesananbst** untuk memasukkan pesanan makanan atau minuman ke dalam keranjang Binary Search Tree.

a. **Memasukkan Node Baru ke dalam Binary Search Tree**

Hasil eksekusi penambahan pesanan makanan sejumlah 8 pesanan.

Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 1 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 2 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 3 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 4 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 5 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 6 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 7 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 8 Pesanan berhasil ditambahkan ke keranjang!	

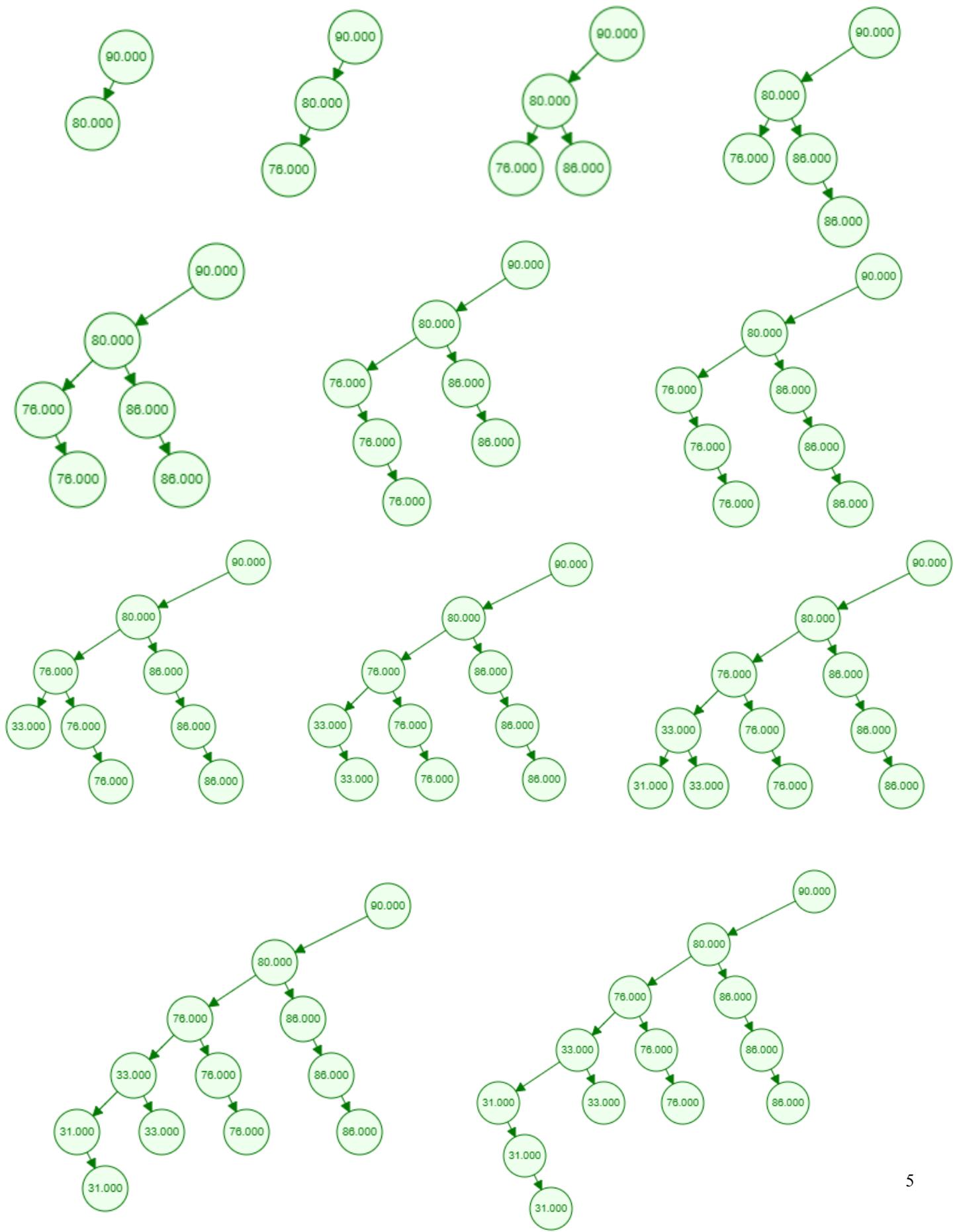
Hasil eksekusi penambahan pesanan minuman sejumlah 8 pesanan.

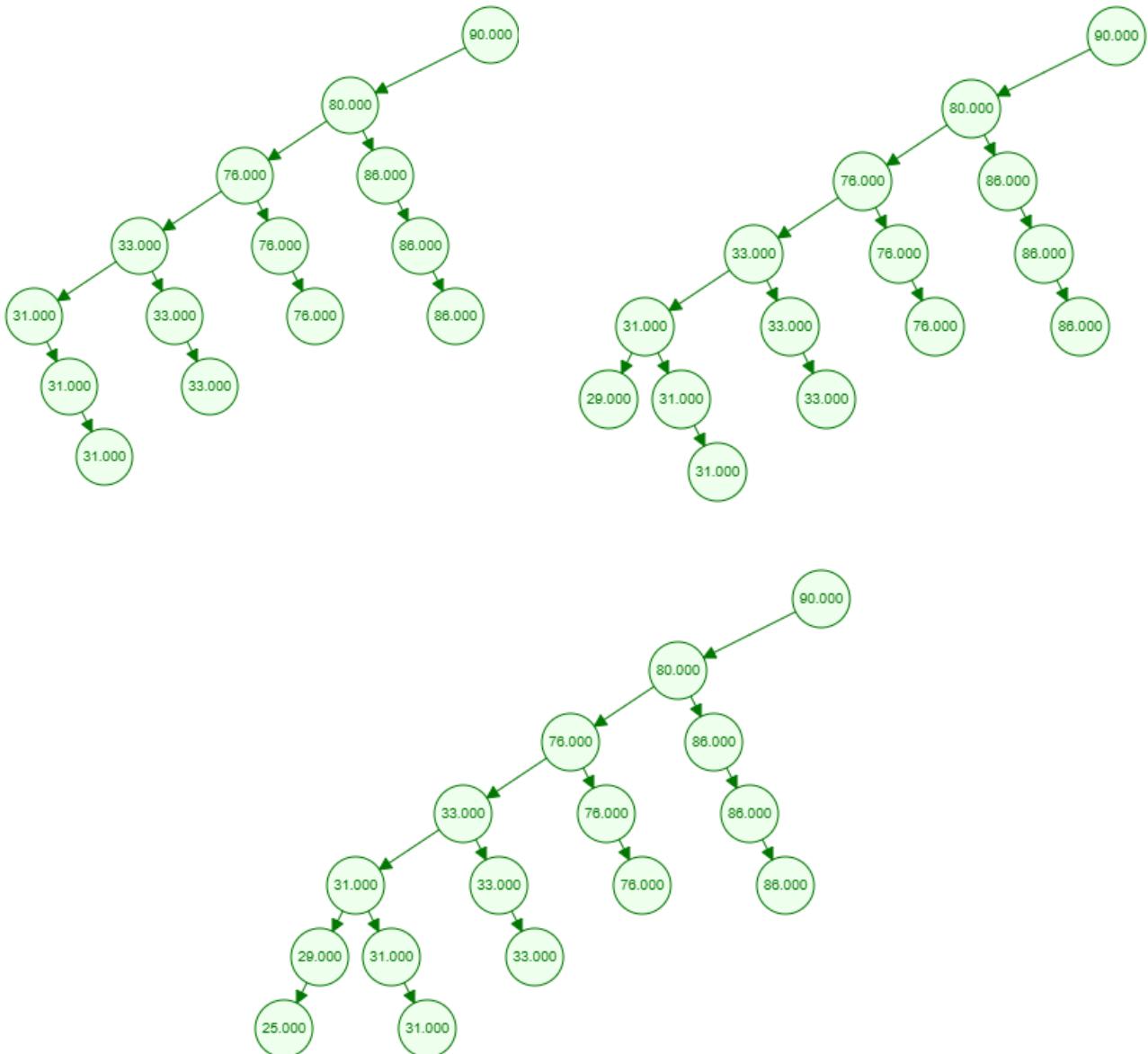
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 4 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 5 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 6 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 7 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 8 Pesanan berhasil ditambahkan ke keranjang!	

Menambahkan 8 pesanan makanan dan 8 pesanan minuman. Berdasarkan hasil eksekusi tersebut, Input:

- Ke-1 menambahkan menu makanan 1, **Splitza Signature**, dengan harga Rp **90.000**.
- Ke-2 menambahkan menu makanan 2, **Splitza Classic Signature**, dengan harga Rp **80.000**.
- Ke-3 menambahkan menu makanan 3, **Veggie Garden**, dengan harga Rp **76.000**.
- Ke-4 menambahkan menu makanan 4, **Meat Lovers**, dengan harga Rp **86.000**.
- Ke-5 menambahkan menu makanan 5, **Super Supreme**, dengan harga Rp **86.000**.
- Ke-6 menambahkan menu makanan 6, **Tuna Melt**, dengan harga Rp **76.000**.
- Ke-7 menambahkan menu makanan 7, **Cheesy Galore**, dengan harga Rp **76.000**.
- Ke-8 menambahkan menu makanan 8, **Meat Lovers Cheesy Mayo**, dengan harga Rp **86.000**.
- Ke-9 menambahkan menu minuman, 1, **Tropical Punch**, dengan harga Rp **33.000**.
- Ke-10 menambahkan menu minuman 2, **Lychee Breeze**, dengan harga Rp **33.000**.
- Ke-11 menambahkan menu minuman 3, **Orange Lychee Sparkle**, dengan harga Rp **31.000**.
- Ke-12 menambahkan menu minuman 4, **Blue Ocean**, dengan harga Rp **31.000**.
- Ke-13 menambahkan menu minuman 5, **Mixberry**, dengan harga Rp **31.000**.
- Ke-14 menambahkan menu minuman 6, **Winter Punch**, dengan harga Rp **33.000**.
- Ke-15 menambahkan menu minuman 7, **Green Tea Yakult**, dengan harga Rp **29.000**.
- Ke-16 menambahkan menu minuman 8, **Honey Lime Tea**, dengan harga Rp **25.000**.

Apabila tanpa Binary Search Tree, maka urutan keranjang akan berdasarkan urutan input. Namun, dengan adanya Binary Search Tree maka urutan keranjang akan berurutan berdasarkan value dari node yaitu harga dari pesanan. Berikut adalah langkah Binary Search Tree dengan 16 nodes dari kiri ke kanan.





```

void inorderTampil(BSTNode* node) {
    if (node == NULL) return;
    inorderTampil(node->left);
    printf("- %s | Harga: Rp. %d\n", node->name, node->harga);
    inorderTampil(node->right);
}
void tampilKeranjangbst(BSTNode* root) {
    if (root == NULL) {
        printf("Keranjang masih kosong.\n");
        return;
    }

    printf("===== Daftar Pesanan di Keranjang =====\n");
    inorderTampil(root);
    printf("===== ======\n");
}

```

Berikut adalah fungsi ***inorderTampil*** dan ***tampilKeranjangbst*** untuk menampilkan keranjang Binary Search Tree dengan Inorder.

```

Masukkan pilihan anda : 4
==== Keranjang Anda ====
===== Daftar Pesanan di Keranjang =====
- Honey Lime Tea | Harga: Rp. 25000
- Green Tea Yakult | Harga: Rp. 29000
- Orange Lychee Sparkle | Harga: Rp. 31000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Winter Punch | Harga: Rp. 33000
- Veggie Garden | Harga: Rp. 76000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Meat Lovers | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Splitza Signature | Harga: Rp. 90000
=====
```

```

BSTNode* hapusPesananInputbst(BSTNode* root) {
    int count = 0;
    int harga, pilihan;
    char daftar[100][50];
    char name[50];

    printf("Masukkan harga pesanan yang ingin dihapus: ");
    scanf("%d", &harga);

    caripesanan(root, harga, daftar, &count);

    if (count == 0) {
        printf("Tidak ditemukan pesanan dengan harga tersebut.\n");
        return root;
    }

    if (count == 1) {
        strcpy(name, daftar[0]);
        printf("Pesanan ditemukan: %s | Harga: %d. Menghapus langsung...\n", name, harga);
    } else {
        printf("Ditemukan %d pesanan dengan harga %d:\n", count, harga);
        for (int i = 0; i < count; i++) {
            printf("%d. %s\n", i + 1, daftar[i]);
        }
        do {
            printf("Pilih nomor pesanan yang ingin dihapus: ");
            scanf("%d", &pilihan);
        } while (pilihan < 1 || pilihan > count);
        strcpy(name, daftar[pilihan - 1]);
    }

    root = hapusPesananbst(root, harga, name);
    printf("Pesanan '%s' dengan harga %d telah dihapus.\n", name, harga);
    return root;
}
```

```

BSTNode* hapusPesananbst(BSTNode* root, int harga, char* name) {
    if (root == NULL) return NULL;

    if (harga < root->harga) {
        root->left = hapusPesananbst(root->left, harga, name);
    } else if (harga > root->harga) {
        root->right = hapusPesananbst(root->right, harga, name);
    } else {
        int cmp = strcmp(root->name, name);
        if (cmp > 0) {
            root->left = hapusPesananbst(root->left, harga, name);
        } else if (cmp < 0) {
            root->right = hapusPesananbst(root->right, harga, name);
        } else {
            if (root->left == NULL) {
                BSTNode* temp = root->right;
                free(root);
                return temp;
            } else if (root->right == NULL) {
                BSTNode* temp = root->left;
                free(root);
                return temp;
            }
            BSTNode* temp = root->right;
            while (temp->left != NULL) temp = temp->left;
            strcpy(root->name, temp->name);
            root->harga = temp->harga;
            root->right = hapusPesananbst(root->right, temp->harga, temp->name);
        }
    }
    return root;
}
```

Berikut adalah hasil eksekusi ketika menu 4. **Tampilkan Keranjang** dijalankan. Dapat dilihat urutan pesanan tidak sesuai urutan input melainkan urutan **In-Order** dari Binary Tree Search yaitu dengan urutan LPR (Left-Print-Right).

Berikut adalah fungsi **hapusPesananInputbst** untuk **input** harga pesanan makanan atau minuman yang ingin dihapus, istilah lainnya adalah node.

Berikut adalah fungsi **hapusPesananbst** untuk mencari dan **menghapus** pesanan makanan atau minuman berdasarkan input harga. Apabila harga node yang ingin dihapus lebih kecil daripada node yang dikunjungi, maka akan ke node kiri dan sebaliknya akan ke kanan. Setelah itu, mengecek apakah memiliki child node.

### b. Melakukan Pencarian Node dan Menghapus Leaf Node pada Binary Search Tree

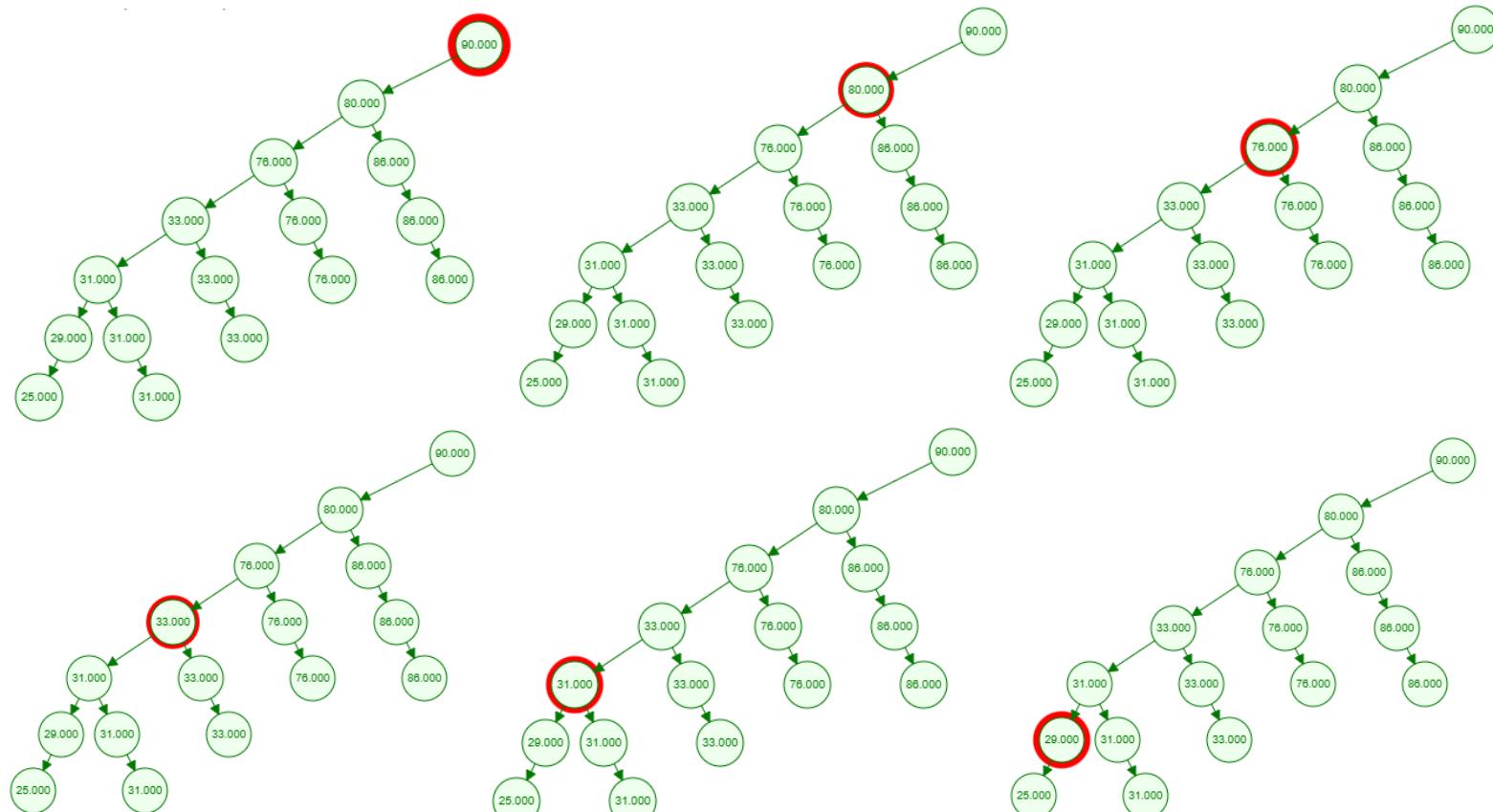
Simulasi akan dilakukan pada node dengan harga Rp 25.000 yaitu menu minuman 8, Honey Lime Tea.

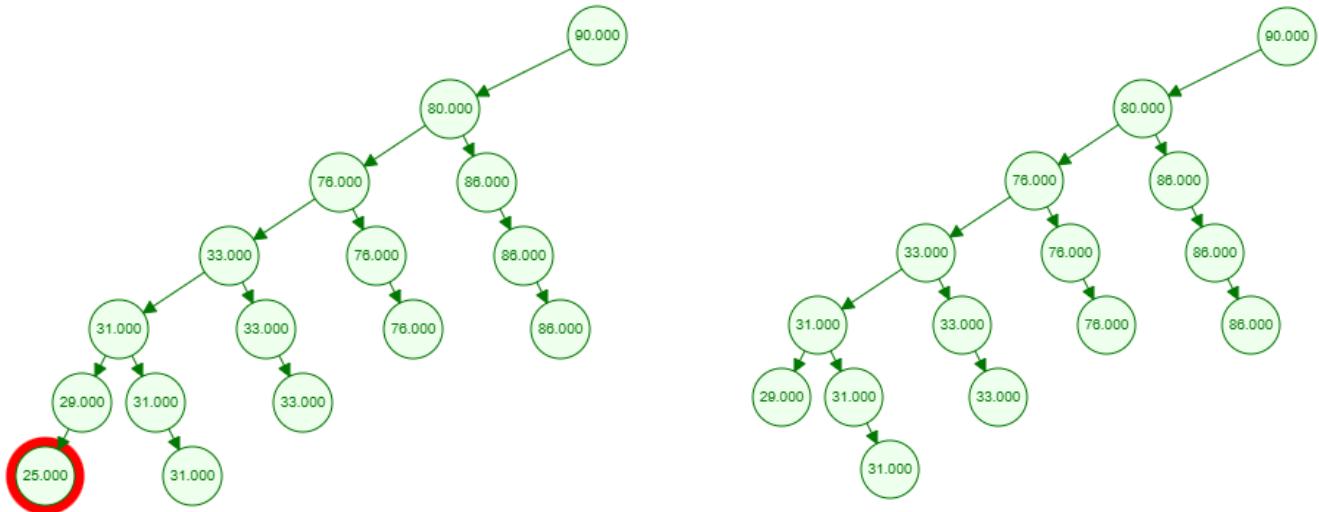
**Masukkan pilihan anda : 3**  
**Masukkan harga pesanan yang ingin dihapus: 25000**  
**Pesanan dengan harga 25000 telah dihapus.**

```
Masukkan pilihan anda : 4
== Keranjang Anda ==
===== Daftar Pesanan di Keranjang =====
- Green Tea Yakult | Harga: Rp. 29000
- Orange Lychee Sparkle | Harga: Rp. 31000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Winter Punch | Harga: Rp. 33000
- Veggie Garden | Harga: Rp. 76000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Meat Lovers | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Splitza Signature | Harga: Rp. 90000
=====
```

Dapat dilihat pada keranjang tersebut, menu **Honey Lime Tea** berhasil ditemukan dan dihapus. Berdasarkan penjelasan sebelumnya, karena node Rp **25.000** merupakan node terkecil dari node lainnya, maka node yang dikunjungi akan berawal dari root dan ke kiri hingga ke ujung. Kemudian karena node tersebut merupakan leaf node atau node yang tidak memiliki child, maka akan langsung di free tanpa melakukan pengecekan child.

Berikut adalah langkah Binary Search Tree dengan 16 nodes dari kiri ke kanan.





### c. Menghapus Node yang Memiliki 1 Child dari Binary Search Tree

Simulasi akan dilakukan pada node dengan harga Rp 29.000 yaitu menu minuman 7, **Green Tea Yakult**.

```

Masukkan pilihan anda : 3
Masukkan harga pesanan yang ingin dihapus: 29000
Pesanan dengan harga 29000 telah dihapus.

```

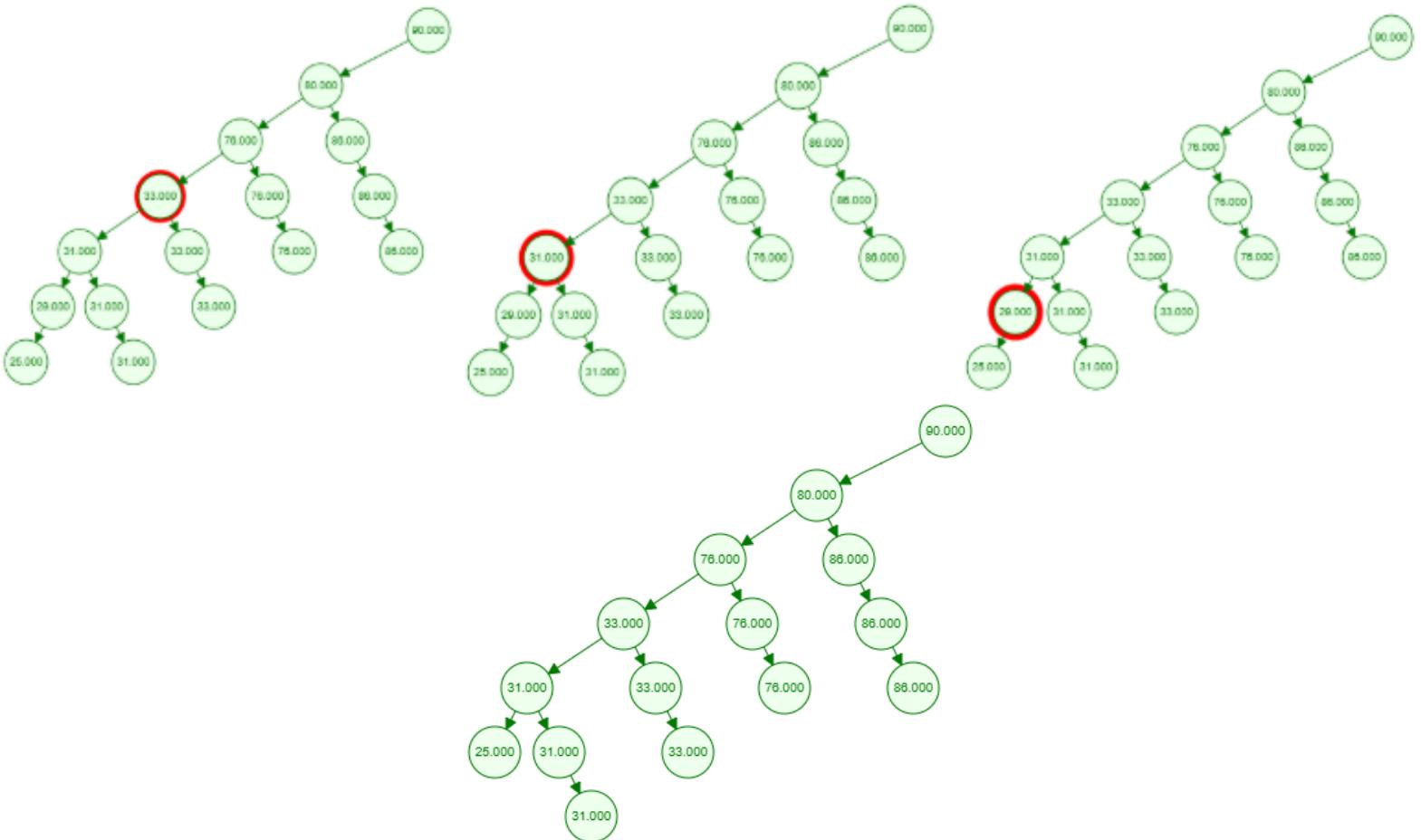
```

Masukkan pilihan anda : 4
== Keranjang Anda ==
===== Daftar Pesanan di Keranjang =====
- Honey Lime Tea | Harga: Rp. 25000
- Orange Lychee Sparkle | Harga: Rp. 31000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Winter Punch | Harga: Rp. 33000
- Veggie Garden | Harga: Rp. 76000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Meat Lovers | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Splitza Signature | Harga: Rp. 90000
=====
```

Dapat dilihat pada keranjang tersebut, menu **Green Tea Yakult** berhasil ditemukan dan dihapus. Berdasarkan penjelasan sebelumnya, karena node Rp **29.000** merupakan node yang memiliki 1 child, maka child node tersebut akan dijadikan **temp** kemudian node yang lama akan di free dan node Rp **29.000** akan diganti dengan **temp** tersebut.

Berikut adalah langkah Binary Search Tree dengan 16 nodes dari kiri ke kanan.





#### d. Menghapus Node yang Memiliki 2 Child dari Binary Search Tree

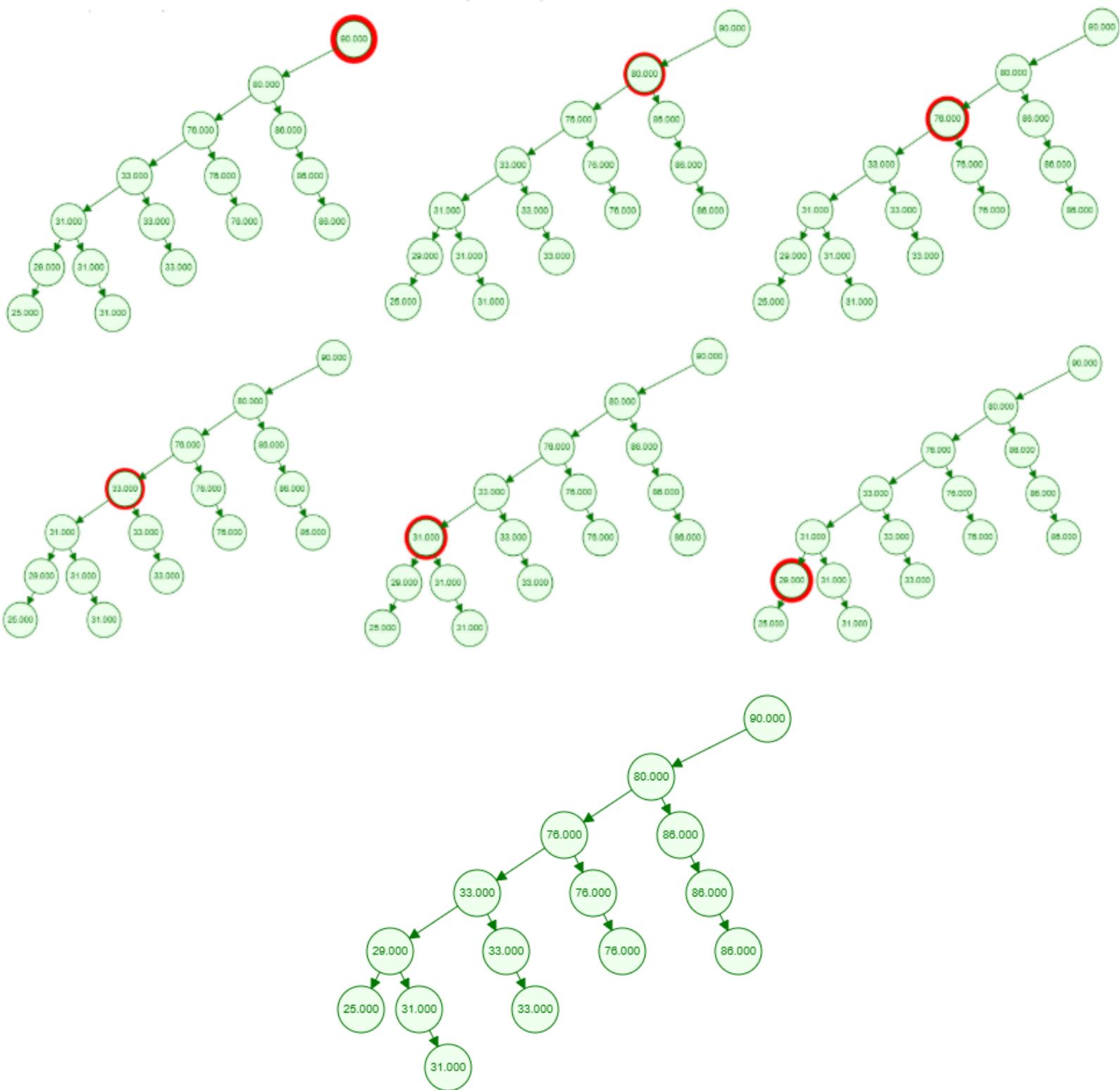
Simulasi akan dilakukan pada node dengan harga Rp 31.000 yaitu menu minuman 3, **Orange Lychee Sparkle**.

```
Masukkan pilihan anda : 3
Masukkan harga pesanan yang ingin dihapus: 31000
Ditemukan 3 pesanan dengan harga 31000:
1. Orange Lychee Sparkle
2. Blue Ocean
3. Mixberry
Pilih nomor pesanan yang ingin dihapus: 1
Pesanan 'Orange Lychee Sparkle' dengan harga 31000 telah dihapus.
```

```
Masukkan pilihan anda : 4
== Keranjang Anda ==
===== Daftar Pesanan di Keranjang ======
- Honey Lime Tea | Harga: Rp. 25000
- Green Tea Yakult | Harga: Rp. 29000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Winter Punch | Harga: Rp. 33000
- Veggie Garden | Harga: Rp. 76000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Meat Lovers | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Splitza Signature | Harga: Rp. 90000
=====
```

Dapat dilihat pada keranjang tersebut, menu **Orange Lychee Sparkle** berhasil ditemukan dan dihapus. Karena node Rp 31.000 memiliki dua child, maka dicari node pengganti (InOrder Successor), yaitu node terkiri pada subtree kanannya. Nilai dari node pengganti tersebut akan menimpapada node Rp 31.000, lalu node pengganti akan dihapus. Karena node pengganti memiliki 1 child, maka proses penghapusan 1-child akan diterapkan pada node tersebut.

Berikut adalah langkah Binary Search Tree dengan 16 nodes dari kiri ke kanan.



## 2.2. Soal 2 : Sub-CPMK0643

Binary Heap kami implementasikan pada jenis order **makanan di ambil**.

```
typedef struct HeapNode{
    char name[100];
    int harga;
} HeapNode;

typedef struct Heap{
    HeapNode data[100];
    int size;
} Heap;
```

```
void insertHeap(Heap* h, menuStruct item) {
    if (h->size >= 100) {
        printf("Keranjang penuh!\n");
        return;
    }

    int i = h->size++;
    while (i > 0 && item.harga > h->data[(i - 1) / 2].harga) {
        h->data[i] = h->data[(i - 1) / 2];
        i = (i - 1) / 2;
    }
    strcpy(h->data[i].name, item.name);
    h->data[i].harga = item.harga;
}
```

Berikut adalah **linked list** untuk Binary Heap sebagai struktur data.

```
void tambahPesananheap(menuStruct makanan[], menuStruct minuman[], Heap* h) {
    int kategori, id;
    printf("Pilih kategori:\n[1] Makanan\n[2] Minuman\n");
    printf("Masukkan pilihan: ");
    scanf("%d", &kategori);

    if (kategori == 1) {
        printf("Masukkan ID makanan: ");
        scanf("%d", &id);
        if (id <= 0 || id > 40) return;
        insertHeap(h, makanan[id - 1]);
    } else if (kategori == 2) {
        printf("Masukkan ID minuman: ");
        scanf("%d", &id);
        if (id <= 0 || id > 15) return;
        insertHeap(h, minuman[id - 1]);
    } else {
        printf("Kategori tidak valid.\n");
    }
}
```

Berikut adalah fungsi **insertHeap** untuk menambahkan node ke dalam Binary Heap Max-Heap dengan harga sebagai value dari node.

Berikut adalah fungsi **tambahPesananheap** untuk memasukkan pesanan makanan atau minuman ke dalam keranjang Binary Heap.

### a. Membuat Binary Heap

Hasil eksekusi penambahan pesanan makanan sejumlah 8 pesanan.

Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 1 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 2 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 3 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 4 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 5 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 6 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 7 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 1 Masukkan ID makanan: 8 Pesanan berhasil ditambahkan ke keranjang!	

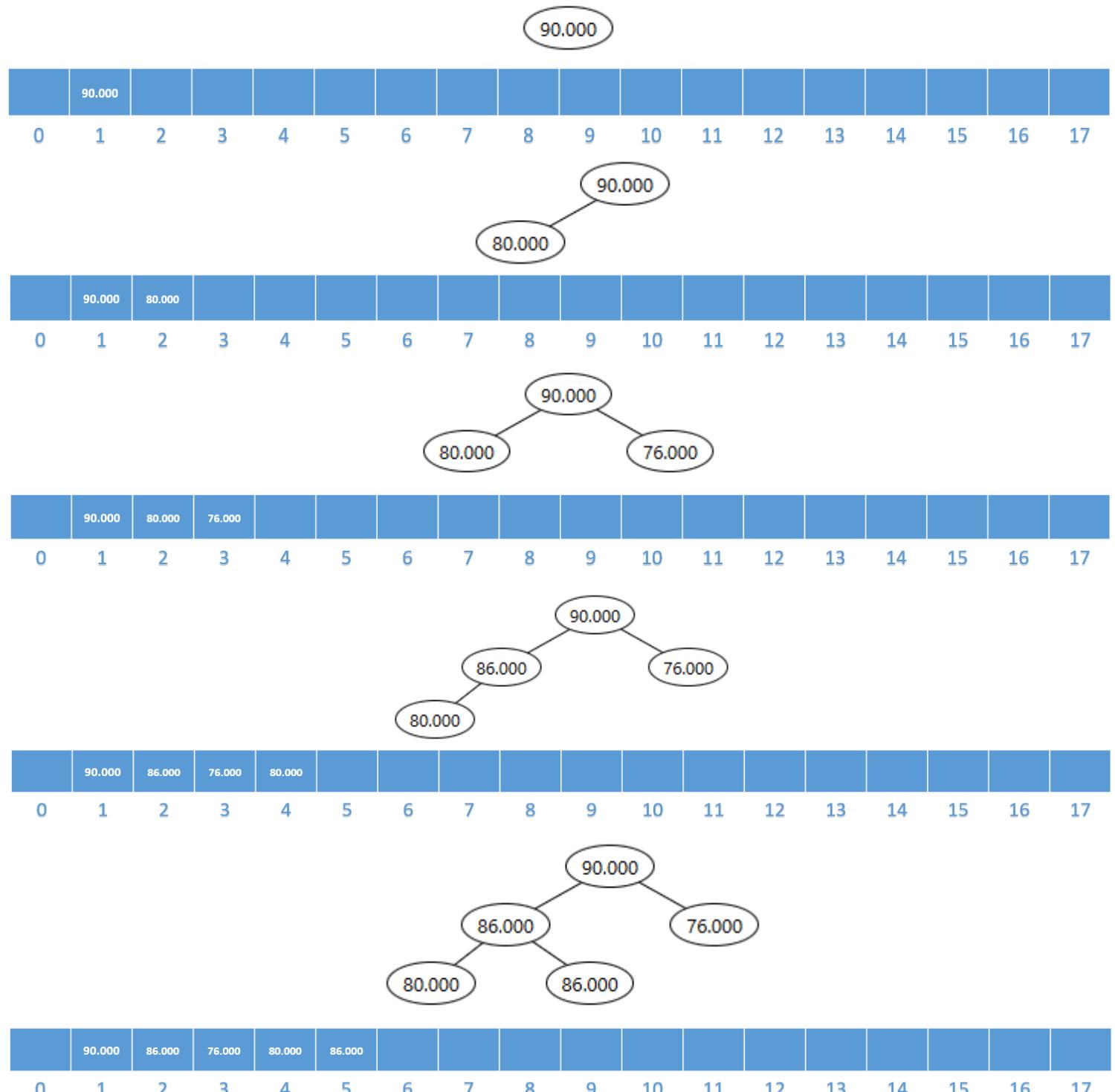
Hasil eksekusi penambahan pesanan minuman sejumlah 8 pesanan.

Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 4 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 5 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 6 Pesanan berhasil ditambahkan ke keranjang!
Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 7 Pesanan berhasil ditambahkan ke keranjang!	Masukkan pilihan anda : 2 Pilih kategori: [1] Makanan [2] Minuman Masukkan pilihan: 2 Masukkan ID minuman: 8 Pesanan berhasil ditambahkan ke keranjang!	

Menambahkan 8 pesanan makanan dan 8 pesanan minuman. Berdasarkan hasil eksekusi tersebut, Input:

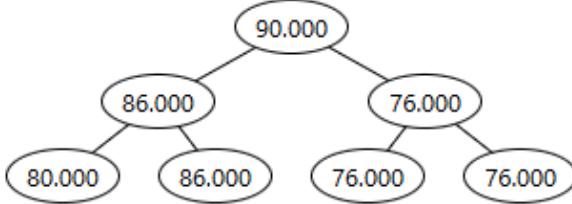
- Ke-1 menambahkan menu makanan 1, **Splitza Signature**, dengan harga Rp **90.000**.
- Ke-2 menambahkan menu makanan 2, **Splitza Classic Signature**, dengan harga Rp **80.000**.
- Ke-3 menambahkan menu makanan 3, **Veggie Garden**, dengan harga Rp **76.000**.
- Ke-4 menambahkan menu makanan 4, **Meat Lovers**, dengan harga Rp **86.000**.
- Ke-5 menambahkan menu makanan 5, **Super Supreme**, dengan harga Rp **86.000**.
- Ke-6 menambahkan menu makanan 6, **Tuna Melt**, dengan harga Rp **76.000**.
- Ke-7 menambahkan menu makanan 7, **Cheesy Galore**, dengan harga Rp **76.000**.
- Ke-8 menambahkan menu makanan 8, **Meat Lovers Cheesy Mayo**, dengan harga Rp **86.000**.
- Ke-9 menambahkan menu minuman, 1, **Tropical Punch**, dengan harga Rp **33.000**.
- Ke-10 menambahkan menu minuman 2, **Lychee Breeze**, dengan harga Rp **33.000**.
- Ke-11 menambahkan menu minuman 3, **Orange Lychee Sparkle**, dengan harga Rp **31.000**.
- Ke-12 menambahkan menu minuman 4, **Blue Ocean**, dengan harga Rp **31.000**.
- Ke-13 menambahkan menu minuman 5, **Mixberry**, dengan harga Rp **31.000**.
- Ke-14 menambahkan menu minuman 6, **Winter Punch**, dengan harga Rp **33.000**.
- Ke-15 menambahkan menu minuman 7, **Green Tea Yakult**, dengan harga Rp **29.000**.
- Ke-16 menambahkan menu minuman 8, **Honey Lime Tea**, dengan harga Rp **25.000**.

Apabila tanpa Binary Heap, maka urutan keranjang akan berdasarkan urutan input. Namun, dengan adanya Binary Heap dengan Max-Heap maka urutan keranjang akan berurutan berdasarkan index dari node yaitu harga dari pesanan. Berikut adalah langkah Binary Heap dengan 16 nodes dari kiri ke kanan.

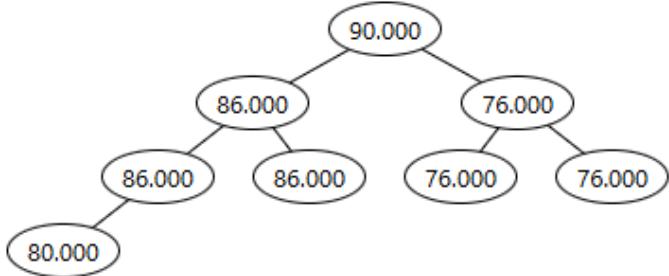




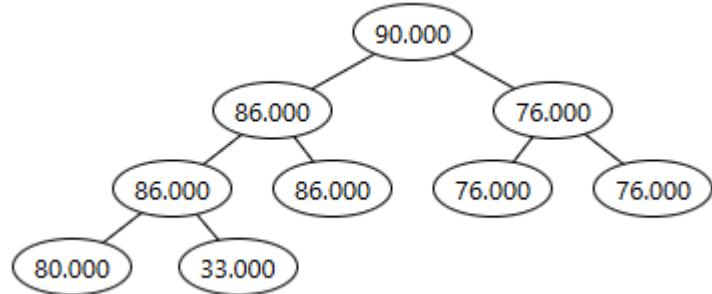
	90.000	86.000	76.000	80.000	86.000	76.000											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



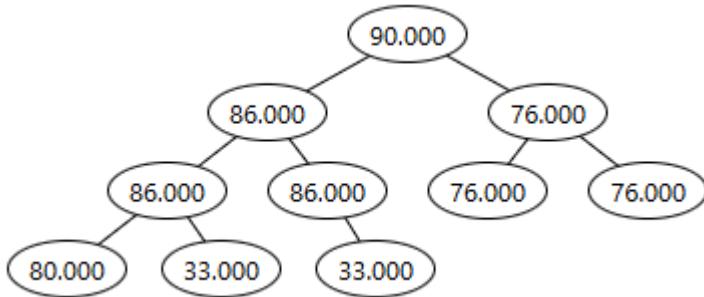
	90.000	86.000	76.000	80.000	86.000	76.000	76.000										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



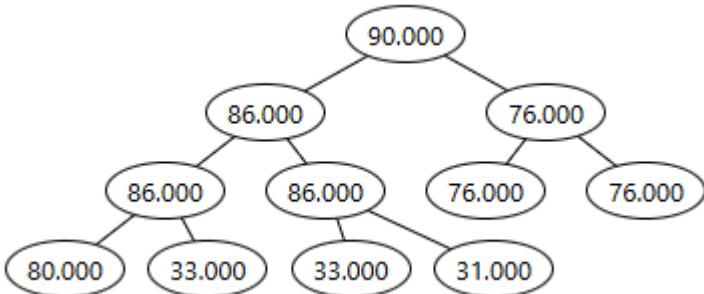
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



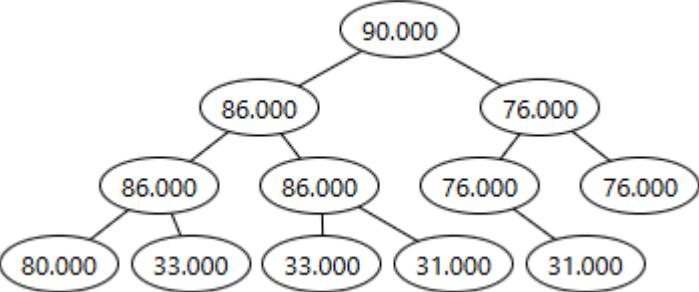
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



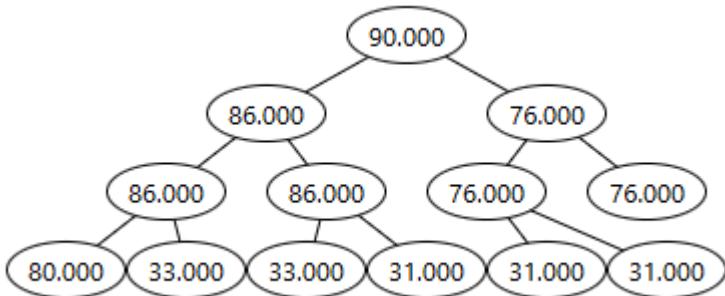
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



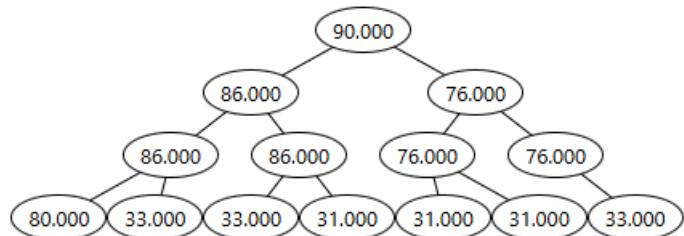
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



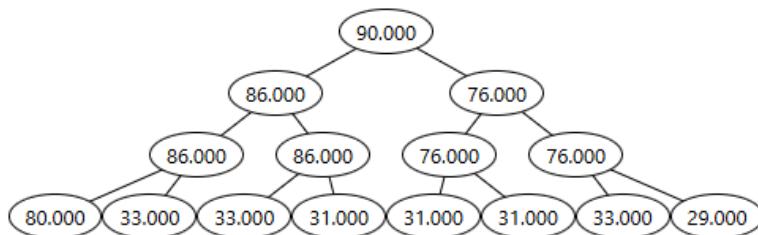
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



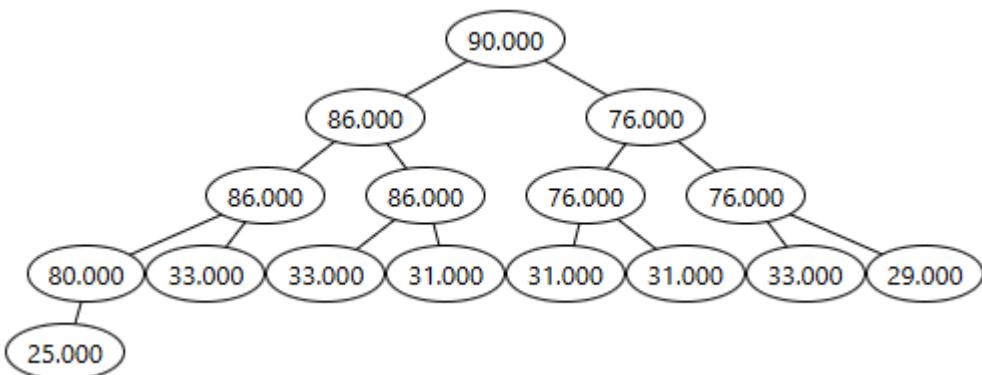
	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000	31.000	33.000		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000	31.000	33.000	29.000	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000	31.000	33.000	29.000	25.000	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	

```
void tampilKeranjangheap(Heap* h) {
    if (h->size == 0) {
        printf("Keranjang kosong.\n");
        return;
    }

    printf("===== Daftar Pesanan (Heap) =====\n");
    for (int i = 0; i < h->size; i++) {
        printf("- %s | Harga: Rp. %d\n", h->data[i].name, h->data[i].harga);
    }
    printf("===== ======\n");
}
```

Berikut adalah fungsi **tampilKeranjangheap** untuk menampilkan keranjang Binary Heap.

```

Masukkan pilihan anda : 4
==== Keranjang Anda ====
===== Daftar Pesanan (Heap) =====
- Splitza Signature | Harga: Rp. 90000
- Meat Lovers | Harga: Rp. 86000
- Veggie Garden | Harga: Rp. 76000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Orange Lychee Sparkle | Harga: Rp. 31000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Winter Punch | Harga: Rp. 33000
- Green Tea Yakult | Harga: Rp. 29000
- Honey Lime Tea | Harga: Rp. 25000
=====
Total harga: Rp. 902000

```

Berikut adalah hasil eksekusi ketika menu 4, **Tampilkan Keranjang** dijalankan. Dapat dilihat bahwa urutan pesanan tidak sesuai dengan urutan input, melainkan mengikuti struktur **Max-Heap** dari **Binary Heap**, yaitu dari kiri ke kanan pada setiap level (tree height - 1).

```

void hapusPesananHeap(Heap* h) {
    int harga, number, pilihan, indeks[100], count = 0;
    char daftar[100][50];
    if (h->size == 0) {printf("Keranjang kosong!\n"); return;}
    printf("Masukkan harga pesanan yang ingin dihapus: "); scanf("%d", &harga);
    for (int i = 0; i < h->size; i++) {
        if (h->data[i].harga == harga) {
            strcpy(daftar[count], h->data[i].name);
            indeks[count] = i;
            count++;
        }
    }
    if (count == 0) {
        printf("Tidak ditemukan pesanan dengan harga tersebut.\n");
        return;
    }
    if (count == 1) {
        number = indeks[0];
        printf("Pesanan ditemukan: %s | Harga: %d. Menghapus langsung...\n", h->data[number].name, h->data[number].harga);
    } else {
        printf("Ditemukan %d pesanan dengan harga %d:\n", count, harga);
        for (int i = 0; i < count; i++) {
            printf("%d. %s\n", i + 1, daftar[i]);
        }
        do {
            printf("Pilih nomor pesanan yang ingin dihapus: ");
            scanf("%d", &pilihan);
        } while (pilihan < 1 || pilihan > count);
        number = indeks[pilihan - 1];
    }
    printf("Menghapus pesanan: %s | Harga: Rp. %d\n", h->data[number].name, h->data[number].harga);
    h->data[number] = h->data[-h->size];
    int i = number;
    while (1) {
        int left = 2 * i + 1, right = 2 * i + 2, largest = i;
        if (left < h->size && h->data[left].harga > h->data[largest].harga) largest = left;
        if (right < h->size && h->data[right].harga > h->data[largest].harga) largest = right;
        if (largest == i) break;
        HeapNode temp = h->data[i];
        h->data[i] = h->data[largest];
        h->data[largest] = temp;
        i = largest;
    }
}

```

Berikut adalah fungsi **hapusPesananHeap** untuk mencari dan menghapus pesanan makanan atau minuman berdasarkan input harga.

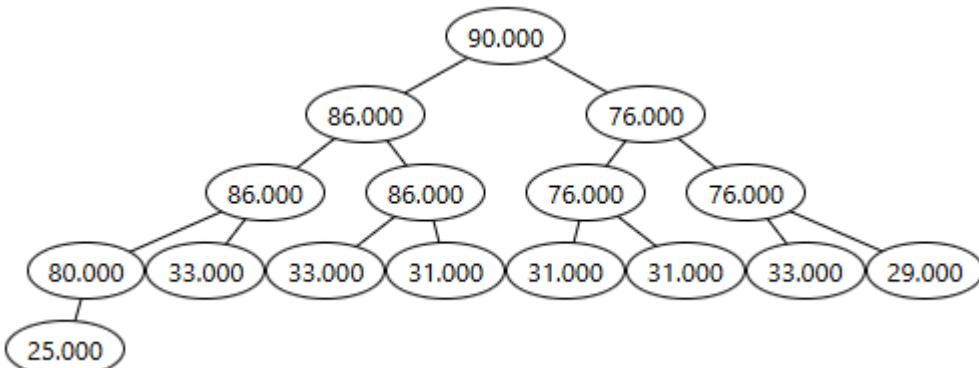
### b. Menghapus Elemen dari Binary Heap

Simulasi akan dilakukan pada node dengan harga Rp **25.000** yaitu menu minuman 3, **Splitza Signature**.

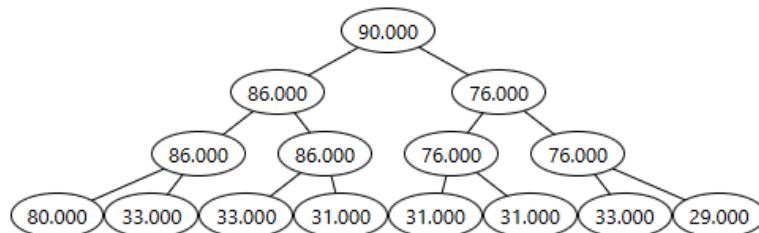
```
Masukkan pilihan anda : 3
Masukkan harga pesanan yang ingin dihapus: 25000
Pesanan ditemukan: Honey Lime Tea | Harga: 25000. Menghapus langsung...
Menghapus pesanan: Honey Lime Tea | Harga: Rp. 25000
```

```
Masukkan pilihan anda : 4
==== Keranjang Anda ====
===== Daftar Pesanan (Heap) =====
- Splitza Signature | Harga: Rp. 90000
- Meat Lovers | Harga: Rp. 86000
- Veggie Garden | Harga: Rp. 76000
- Meat Lovers Cheesy Mayo | Harga: Rp. 86000
- Super Supreme | Harga: Rp. 86000
- Tuna Melt | Harga: Rp. 76000
- Cheesy Galore | Harga: Rp. 76000
- Splitza Classic | Harga: Rp. 80000
- Tropical Punch | Harga: Rp. 33000
- Lychee Breeze | Harga: Rp. 33000
- Orange Lychee Sparkle | Harga: Rp. 31000
- Blue Ocean | Harga: Rp. 31000
- Mixberry | Harga: Rp. 31000
- Winter Punch | Harga: Rp. 33000
- Green Tea Yakult | Harga: Rp. 29000
=====
Total harga: Rp. 877000
```

Dapat dilihat pada keranjang tersebut, menu **Splitza Signature** berhasil ditemukan dan dihapus. Karena node Rp **25.000** merupakan node dengan harga terkecil dan berada di ujung array maka langsung di free saja. Pengecekan dimulai dari root dan linear search.



	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000	33.000	29.000	25.000	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17



	90.000	86.000	76.000	86.000	86.000	76.000	76.000	80.000	33.000	33.000	31.000	31.000	31.000	33.000	29.000		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

### 2.3. Soal 3 : Sub-CPMK0644

Menu yang disediakan sudah disiapkan oleh fungsi **menuMakanan** lalu ditampilkan, setelah itu akan ada opsi untuk memilih antara Bubble sort(Descending) atau Insertion sort(Ascending)

```
// Menginisialisasi menu makanan dan minuman dengan file processing
void menuMakanan(menuStruct makanan[], menuStruct minuman[])
{
    // Inisialisasi menu makanan
    int counter = 0;
    FILE *fp = fopen("menuMakanan.txt", "r");
    if (!fp)
    {
        printf("Error: Tidak bisa membuka file menuMakanan.txt\n");
        return;
    }
    while (counter < 40 && fscanf(fp, "%[^#]#%d\n", &makanan[counter].name, &makanan[counter].harga) == 2)
    {
        counter++;
    }
    fclose(fp);

    // Inisialisasi menu minuman
    counter = 0;
    fp = fopen("menuMinuman.txt", "r");
    if (!fp)
    {
        printf("Error: Tidak bisa membuka file menuMinuman.txt\n");
        return;
    }
    while (counter < 15 && fscanf(fp, "%[^#]#%d\n", &minuman[counter].name, &minuman[counter].harga) == 2)
    {
        counter++;
    }
    fclose(fp);
}
```

Tampilan opsi sorting :

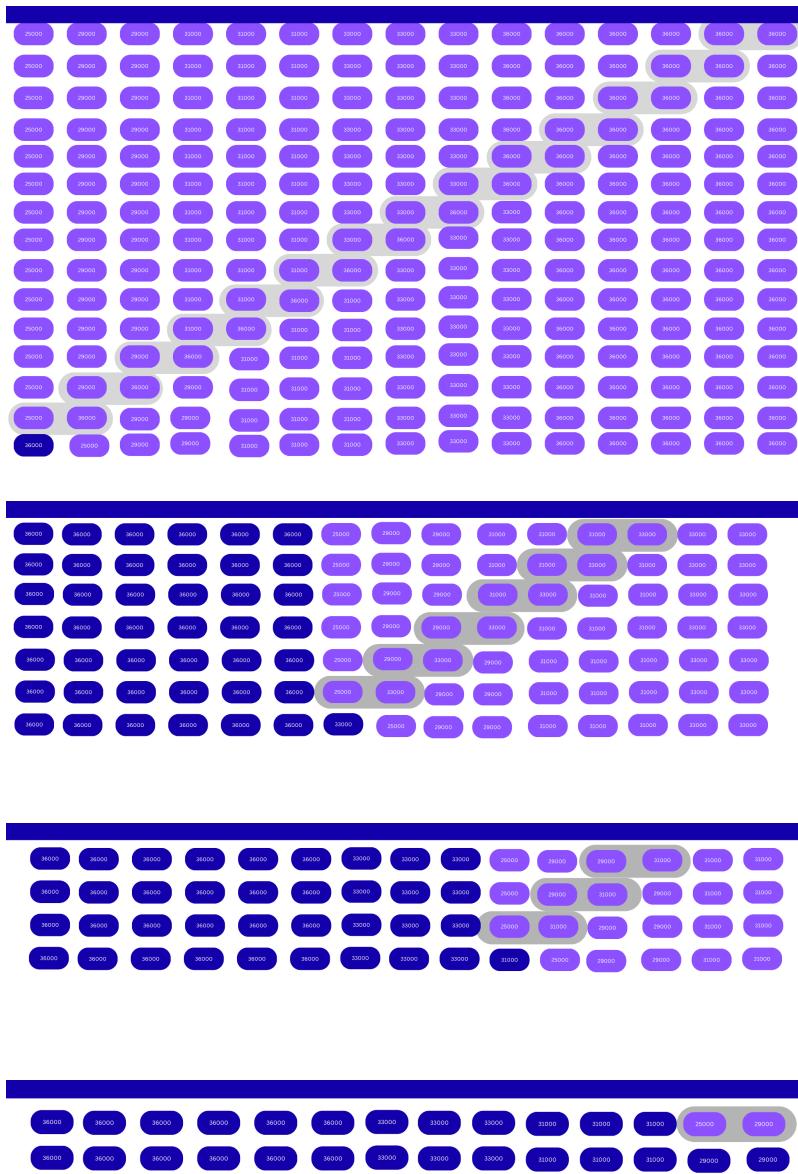
```
== Menu Sorting dan Searching ==
1. Bubble Sort (Descending)
2. Insertion Sort (Ascending)
3. Binary Search (berdasarkan harga)
4. Kembali ke menu utama
Pilihan Anda:
```

Berikut adalah fungsi **bubbleSortMenu** yang menampilkan menu secara Descending dan **insertionSortMenu** menampilkan menu secara Ascending :

```
// Bubble sort dari harga terbesar ke terkecil
void bubbleSortMenu(menuStruct arr[], int n){
    int i, j;
    for (i = 1; i < n; i++) {
        for (j = n - 1; j > 0; j--) {
            if (arr[j].harga > arr[j - 1].harga) {
                swapStruct(&arr[j], &arr[j - 1]);
            }
        }
    }
}

// Insertion sort dari harga terkecil ke terbesar
void insertionSortMenu(menuStruct arr[], int n){
    int i, j;
    menuStruct temp;
    for (i = 0; i < n; i++) {
        temp = arr[i];
        for (j = i - 1; j >= 0 && arr[j].harga > temp.harga; j--) {
            arr[j + 1] = arr[j];
        }
        arr[j + 1] = temp;
    }
}
```

### Ilustrasi Bubble sort (Desc):



## Hasil Bubble sort :

```

1. Bubble Sort (Descending)
2. Insertion Sort (Ascending)
3. Binary Search (berdasarkan harga)
4. Kembali ke menu utama
Pilihan Anda: 1
Menu diurutkan dari harga tertinggi ke terendah.
  
```

Menampilkan Menu Makanan			Menampilkan Menu Minuman		
ID	Nama Makanan	Harga	ID	Nama Minuman	Harga
1	Splitza Signature	Rp. 90000	1	Watermelon Juice	Rp. 36000
2	Meat Lovers	Rp. 86000	2	Melon Juice	Rp. 36000
3	Super Supreme	Rp. 86000	3	Strawberry Watermelon Juice	Rp. 36000
4	Meat Lovers Cheesy Mayo	Rp. 86000	4	Strawberry Orange Juice	Rp. 36000
5	American Favourite	Rp. 86000	5	Melon Peach Juice	Rp. 36000
6	Super Supreme Chicken	Rp. 86000	6	Avocado Juice	Rp. 36000
7	Pepperoni	Rp. 86000	7	Tropical Punch	Rp. 33000
8	Chicken Puff Pizza	Rp. 86000	8	Lychee Breeze	Rp. 33000
9	Meat Lovers Chicken	Rp. 86000	9	Winter Punch	Rp. 33000
10	Splitza Classic	Rp. 80000	10	Orange Lychee Sparkle	Rp. 31000
11	Veggie Garden	Rp. 76000	11	Blue Ocean	Rp. 31000
12	Tuna Melt	Rp. 76000	12	Mixberry	Rp. 31000
13	Cheesy Galore	Rp. 76000	13	Green Tea Yakult	Rp. 29000
14	Meaty	Rp. 76000	14	Lychee Spring	Rp. 29000
15	Hawaiian Chicken	Rp. 76000	15	Honey Lime Tea	Rp. 25000
16	Cumi Cabe Ijo	Rp. 73000			
17	Oriental Chicken Spaghetti	Rp. 66000			
18	Beef Lasagna	Rp. 65000			
19	Chicken Fettuccine alla Italia	Rp. 65000			
20	Black Pepper Chicken/Beef Fettuccine	Rp. 54000			
21	Tuna Aglio Olio Spaghetti	Rp. 54000			
22	Beef Spaghetti	Rp. 46000			
23	Meatballs Beef Mushroom Rice	Rp. 43000			
24	Black Pepper Chicken Rice	Rp. 43000			
25	Oriental Chicken Rice	Rp. 43000			
26	Thai Chicken Rice	Rp. 43000			
27	New Orleans Chicken Wings	Rp. 43000			
28	Baked Chicken Chunks	Rp. 42000			
29	Fresh Salad	Rp. 42000			
30	Chicken Royale	Rp. 41000			
31	Cheese Martabak Pizza	Rp. 37000			
32	Sausage Pastry Roll	Rp. 34000			
33	Nacho Cheese	Rp. 34000			
34	Choco Puff	Rp. 34000			
35	Cheese Rolls	Rp. 34000			
36	Deluxe Beef Bruschetta	Rp. 28000			
37	Deluxe Chicken Bruschetta	Rp. 28000			
38	Garlic Cheese Bread	Rp. 28000			
39	Potato Wedges	Rp. 26000			
40	Garlic Bread	Rp. 24000			

## Ilustrasi insertion sort(Asce) :



## Hasil Insertion sort :

Menampilkan Menu Makanan			Menampilkan Menu Minuman		
ID	Nama Makanan	Harga	ID	Nama Minuman	Harga
1	Garlic Bread	Rp. 24000	1	Honey Lime Tea	Rp. 25000
2	Potato Wedges	Rp. 26000	2	Green Tea Yakult	Rp. 29000
3	Deluxe Beef Bruschetta	Rp. 28000	3	Lychee Spring	Rp. 29000
4	Deluxe Chicken Bruschetta	Rp. 28000	4	Orange Lychee Sparkle	Rp. 31000
5	Garlic Cheese Bread	Rp. 28000	5	Blue Ocean	Rp. 31000
6	Sausage Pastry Roll	Rp. 34000	6	Mixberry	Rp. 31000
7	Nacho Cheese	Rp. 34000	7	Tropical Punch	Rp. 33000
8	Choco Puff	Rp. 34000	8	Lychee Breeze	Rp. 33000
9	Cheese Rolls	Rp. 34000	9	Winter Punch	Rp. 33000
10	Cheese Martabak Pizza	Rp. 37000	10	Watermelon Juice	Rp. 36000
11	Chicken Royale	Rp. 41000	11	Melon Juice	Rp. 36000
12	Baked Chicken Chunks	Rp. 42000	12	Strawberry Watermelon Juice	Rp. 36000
13	Fresh Salad	Rp. 42000	13	Strawberry Orange Juice	Rp. 36000
14	Meatballs Beef Mushroom Rice	Rp. 43000	14	Melon Peach Juice	Rp. 36000
15	Black Pepper Chicken Rice	Rp. 43000	15	Avocado Juice	Rp. 36000
16	Oriental Chicken Rice	Rp. 43000			
17	Thai Chicken Rice	Rp. 43000			
18	New Orleans Chicken Wings	Rp. 43000			
19	Beef Spaghetti	Rp. 46000			
20	Black Pepper Chicken/Beef Fettuccine	Rp. 54000			
21	Tuna Aglio Olio Spaghetti	Rp. 54000			
22	Beef Lasagna	Rp. 65000			
23	Chicken Fettuccine alla Italia	Rp. 65000			
24	Oriental Chicken Spaghetti	Rp. 66000			
25	Cumi Cabe Ijo	Rp. 73000			
26	Veggie Garden	Rp. 76000			
27	Tuna Melt	Rp. 76000			
28	Cheesy Galore	Rp. 76000			
29	Meaty	Rp. 76000			
30	Hawaiian Chicken	Rp. 76000			
31	Splitza Classic	Rp. 80000			
32	Meat Lovers	Rp. 86000			
33	Super Supreme	Rp. 86000			
34	Meat Lovers Cheesy Mayo	Rp. 86000			
35	American Favourite	Rp. 86000			
36	Super Supreme Chicken	Rp. 86000			
37	Pepperoni	Rp. 86000			
38	Chicken Puff Pizza	Rp. 86000			
39	Meat Lovers Chicken	Rp. 86000			
40	Splitza Signature	Rp. 90000			

## 2.4. Soal 4 : Sub-CPMK0644

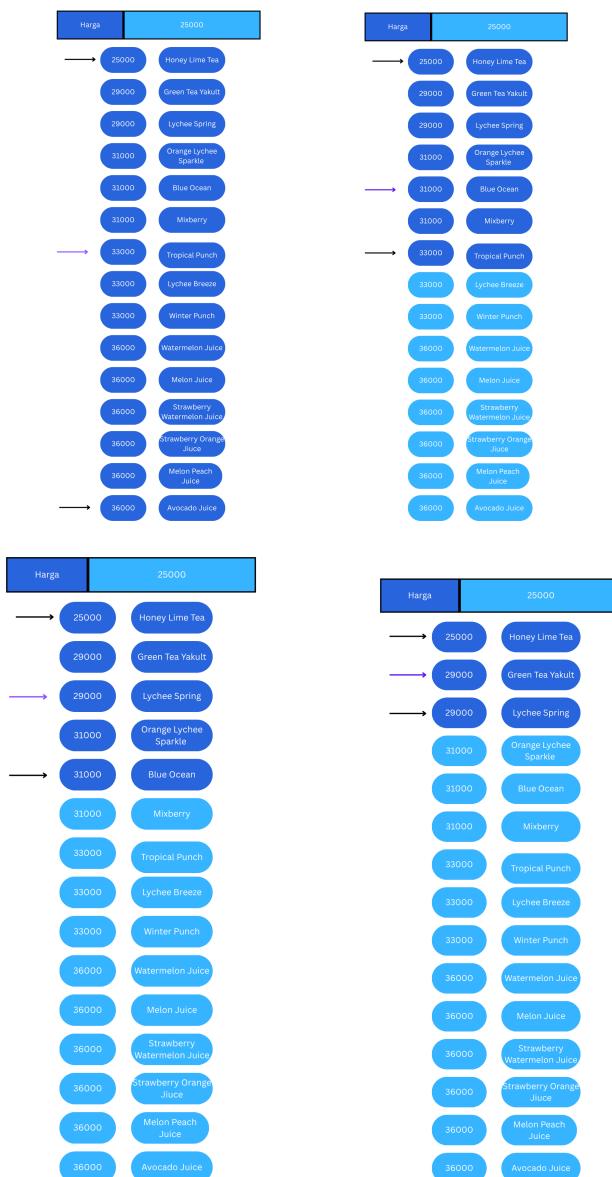
Binary Search diimplementasikan untuk mencari Menu makanan atau minuman yang diambil dari sorting sebelumnya dengan harga tertentu

```
int binarySearch(menuStruct arr[], int n, int target, int isAscending) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = (left + right) / 2;
        if (arr[mid].harga == target)
            return mid;
        if (isAscending) {
            if (arr[mid].harga < target)
                left = mid + 1;
            else
                right = mid - 1;
        } else {
            if (arr[mid].harga > target)
                left = mid + 1;
            else
                right = mid - 1;
        }
    }
    return -1;
}

case 3:
if (*sortedFlag == 0) {
    printf("Anda harus melakukan sorting terlebih dahulu!\n");
} else {
    int kategori, harga, index;
    printf("Pilih kategori: Makanan (1) atau Minuman (2)\nMasukkan pilihan anda: ");
    scanf("%d", &kategori);
    printf("Masukkan harga yang ingin dicari: ");
    scanf("%d", &harga);

    if (kategori == 1) {
        index = binarySearch(makanan, 40, harga, *sortedFlag == 2);
        if (index != -1)
            printf("Ditemukan:[%d] %s | Harga: Rp. %d\n", index+1, makanan[index].name, makanan[index].harga);
        else
            printf("Tidak ditemukan makanan dengan harga Rp. %d\n", harga);
    } else if (kategori == 2) {
        index = binarySearch(minuman, 15, harga, *sortedFlag == 2);
        if (index != -1)
            printf("Ditemukan:[%d] %s | Harga: Rp. %d\n", index+1, minuman[index].name, minuman[index].harga);
        else
            printf("Tidak ditemukan minuman dengan harga Rp. %d\n", harga);
    } else {
        printf("Kategori tidak valid.\n");
    }
}
break;
```

## Ilustrasi Binary search :





Hasil Binary Search :

```
== Menu Sorting dan Searching ==
1. Bubble Sort (Descending)
2. Insertion Sort (Ascending)
3. Binary Search (berdasarkan harga)
4. Kembali ke menu utama
Pilihan Anda: 3
Pilih kategori: Makanan (1) atau Minuman (2)
Masukkan pilihan anda: 1
Masukkan harga yang ingin dicari: 86000
Ditemukan:[35] American Favourite | Harga: Rp. 86000
```

Apabila pengguna memasukan harga yang tidak sesuai dengan harga yang ada di menu maka akan ada pemberitahuan bahwa menu tersebut tidak ada

```
== Menu Sorting dan Searching ==
1. Bubble Sort (Descending)
2. Insertion Sort (Ascending)
3. Binary Search (berdasarkan harga)
4. Kembali ke menu utama
Pilihan Anda: 3
Pilih kategori: Makanan (1) atau Minuman (2)
Masukkan pilihan anda: 1
Masukkan harga yang ingin dicari: 40000
Tidak ditemukan makanan dengan harga Rp. 40000
```

Atau jika pengguna belum melakukan sorting sama sekali maka Binary search tidak dapat dilakukan

```
== Menu Sorting dan Searching ==
1. Bubble Sort (Descending)
2. Insertion Sort (Ascending)
3. Binary Search (berdasarkan harga)
4. Kembali ke menu utama
Pilihan Anda: 3
Anda harus melakukan sorting terlebih dahulu!
```

## **BAB 3**

### **PEMBAGIAN**

### **TUGAS**

#### **3.1 Ketua Kelompok**

Kevin Mikael

- Mengerjakan Soal 1: Sub-CPMK0643
- Mengerjakan Soal 2: Sub-CPMK0643

#### **3.2 Anggota Kelompok**

Adam Rifqy Hajat

- Mengerjakan Soal 3: Sub-CPMK0644
- Mengerjakan Soal 4: Sub-CPMK0644

Bagus Kuncoro

- Menyusun Laporan Bab 2.1. Soal 3: Sub-CPMK0644
- Menyusun Laporan Bab 2.1. Soal 4: Sub-CPMK0644

Christian Surya T

- Menyusun Laporan Bab 1. Pendahuluan
- Menyusun Laporan Bab 2.1. Soal 1: Sub-CPMK0643
- Menyusun Laporan Bab 2.2. Soal 2: Sub-CPMK0643
- Menyusun Laporan Bab 5. Kesimpulan

## **BAB 4**

### **EVALUASI**

#### **4.1 Ketua Kelompok**

Kevin Mikael

- UAS ini sangatlah menantang dan menyenangkan karena mengimplementasikan banyak algoritma-algoritma seperti bst dan heap, saya juga senang berada di kelompok ini dikarenakan sangat aktifnya dalam berkomunikasi

#### **4.2 Anggota Kelompok**

Adam Rifqy Hajat

- Sangat seru untuk mengimplementasi materi materi setelah UTS ke dalam tugas ini. Karena kita hanya menambahkan algoritma-algoritma baru di atas tugas UTS kita yang sebelumnya. Ini membuat saya lebih paham dan intim kepada bab-bab algoritma UAS.

Bagus Kuncoro

- UAS kali ini bisa dikatakan cukup melelahkan terutama didalam pembuatan laporan karena didalam soal diminta untuk membuat gambar dari proses suatu fungsi yang memakan waktu cukup lama, tetapi saya menjadi paham tentang materi yang dikerjakan

Christian Surya T

- Menurut saya soal UAS ini sangat menantang kemudian jumlah soal dan kerjaan yang perlu dilakukan suatu kelompok juga sudah cocok, dua anggota mengerjakan tugas praktek dan dua anggota lainnya mengerjakan tugas laporan atau teorinya. Dengan pembagian tersebut, setiap anggota dapat saling belajar. Contohnya saya dapat belajar tugas praktek dan menghubungkannya dengan teori-teori atau visualisasi. Kelompok sangat komunikatif dan pengeraaan dapat berjalan dengan lancar.

## **BAB 5**

### **KESIMPULAN**

Aplikasi *Pizza Hut Food and Drink Ordering* yang kami kembangkan menggunakan bahasa pemrograman C merupakan implementasi sistem pemesanan makanan dan minuman yang sederhana namun efektif. Aplikasi ini dirancang untuk memberikan pengalaman pemesanan yang menyerupai proses di restoran cepat saji, dengan mengedepankan kemudahan penggunaan, kecepatan interaksi, serta pengelolaan data yang efisien.

Berbagai struktur data dan algoritma memiliki peran spesifik dalam sistem, BST efektif untuk menyimpan dan mengelola isi keranjang berdasarkan nama menu, memungkinkan pencarian dan penghapusan node secara efisien. Binary Max-Heap digunakan untuk menampilkan keranjang berdasarkan urutan harga tertinggi ke terendah. Bubble Sort dan Insertion Sort berhasil digunakan untuk mengurutkan daftar menu berdasarkan harga, meskipun efisiensinya terbatas pada jumlah data kecil hingga menengah. Binary Search mempermudah pencarian cepat menu setelah data diurutkan.

Program dapat menampilkan isi keranjang dalam bentuk heap array yang mencerminkan urutan Max-Heap, bukan urutan input. Kemudian Penghapusan pesanan dari keranjang dilakukan berdasarkan harga, dan jika node yang dihapus adalah elemen dengan harga terendah di ujung array, maka cukup dilakukan free tanpa perlu heapify. Secara keseluruhan, pemilihan struktur data dan algoritma yang tepat sangat bergantung pada sifat data dan jenis operasi yang paling sering dilakukan pada data tersebut.

Dengan demikian, aplikasi ini telah memenuhi tujuan utama pengembangan, yaitu menciptakan sistem pemesanan yang interaktif dan edukatif, yang dapat digunakan sebagai referensi dalam pengembangan aplikasi pemesanan makanan digital berbasis teks.