

**LAPORAN UJIAN AKHIR SEMESTER ALGORITMA DAN
STRUKTUR DATA: PROGRAM MANAJEMEN
PERPUSTAKAAN PRIBADI**



UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Anggota Kelompok

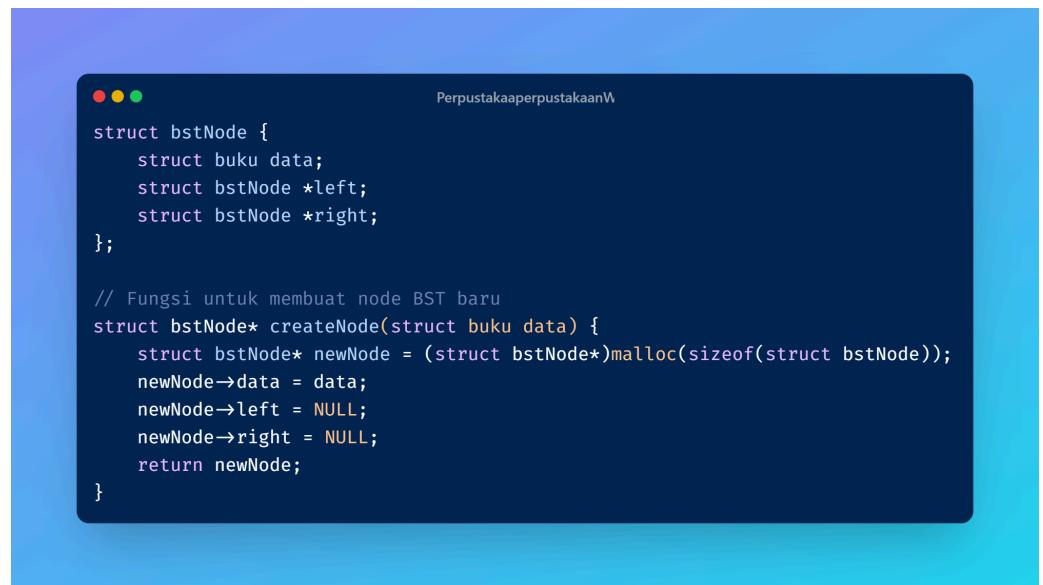
- 1. Gading Kelana Putra (00000111323)**
- 2. Naufal Rabbani (00000108931)**
- 3. Rafael Romelo Gibran (00000111248)**
- 4. Felix Fernando Williams Lim (00000108757)**

1. Implementasi penggunaan Binary Search Tree

a. Membaca Data

Dalam Program sistem manajemen perpustakaan, file processing dengan mode akses read digunakan pada fungsi:

- i. `createNode()` digunakan untuk membuat node baru untuk BST dengan cara mengalokasikan memory lalu akan mengisi datanya ke struct buku.

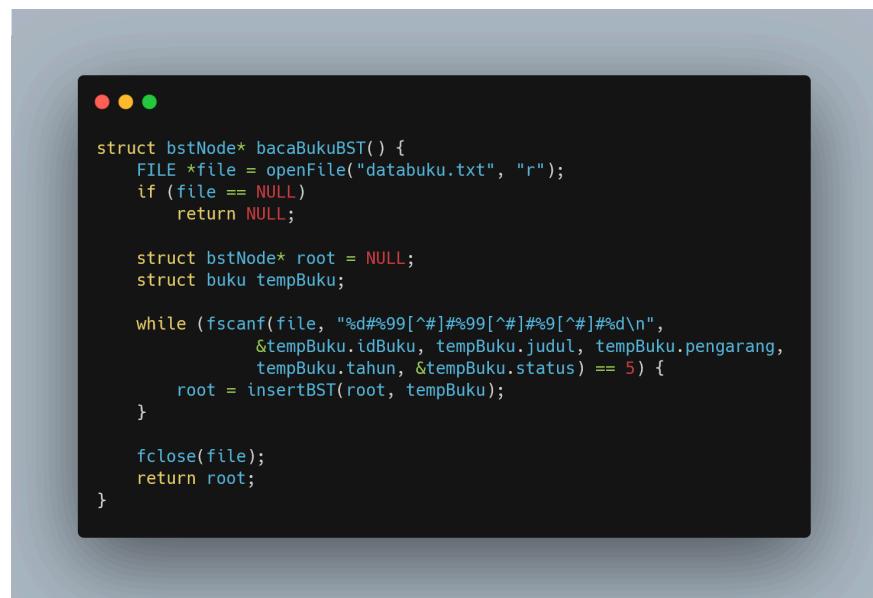


```
PerpustakaaperpustakaanW

struct bstNode {
    struct buku data;
    struct bstNode *left;
    struct bstNode *right;
};

// Fungsi untuk membuat node BST baru
struct bstNode* createNode(struct buku data) {
    struct bstNode* newNode = (struct bstNode*)malloc(sizeof(struct bstNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
```

- ii. `bacaBukuBST()` digunakan untuk membaca list buku dalam database file `databuku.txt` untuk keperluan membuat tree dalam program.



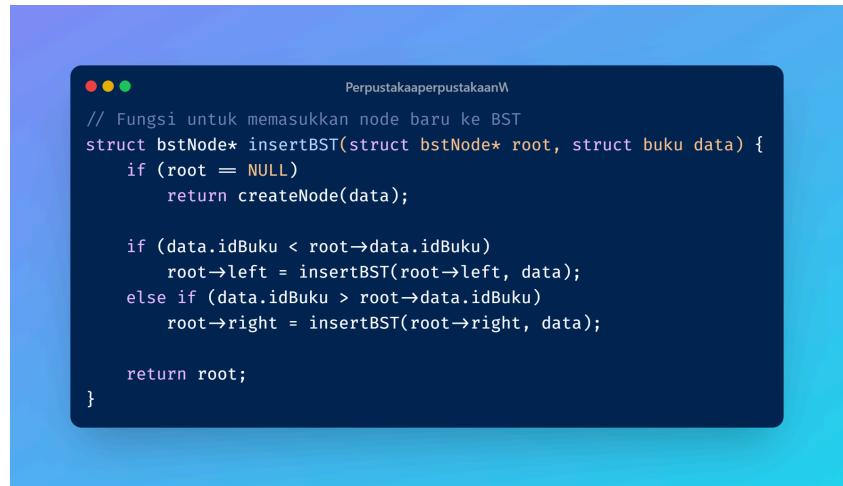
```
struct bstNode* bacaBukuBST() {
    FILE *file = fopen("databuku.txt", "r");
    if (file == NULL)
        return NULL;

    struct bstNode* root = NULL;
    struct buku tempBuku;

    while (fscanf(file, "%d%99[^#]%99[^#]#%99[^#]#%d\n",
                  &tempBuku.idBuku, tempBuku.judul, tempBuku.pengarang,
                  tempBuku.tahun, &tempBuku.status) == 5) {
        root = insertBST(root, tempBuku);
    }

    fclose(file);
    return root;
}
```

- iii. insertBST() digunakan setelah fungsi bacaBukuBST() berhasil membaca buku yang ada di databuku.txt. Fungsi ini berguna untuk mengatur siapa yang bakal menjadi left child dan right child agar bisa menjadi tree yang benar.



```
● ● ● PerpustakaanperpustakaanW
// Fungsi untuk memasukkan node baru ke BST
struct bstNode* insertBST(struct bstNode* root, struct buku data) {
    if (root == NULL)
        return createNode(data);

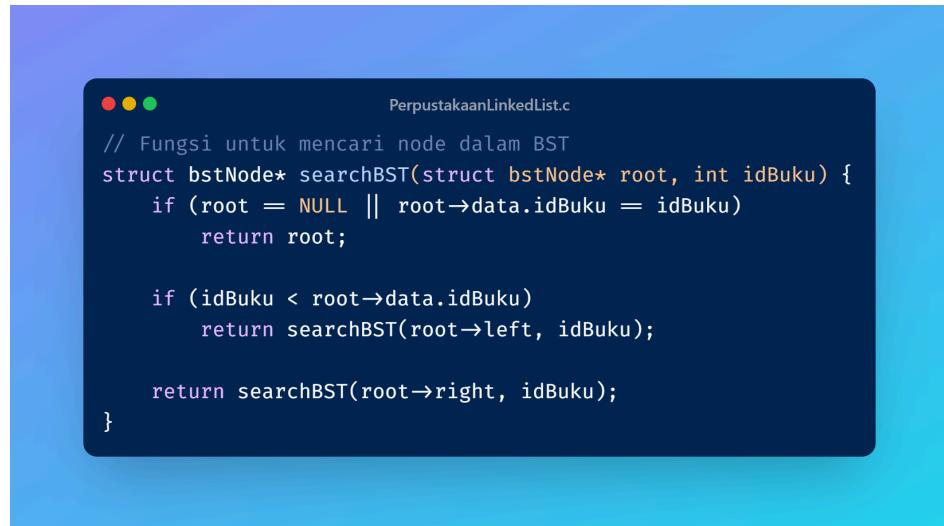
    if (data.idBuku < root->data.idBuku)
        root->left = insertBST(root->left, data);
    else if (data.idBuku > root->data.idBuku)
        root->right = insertBST(root->right, data);

    return root;
}
```

b. Mencari node

Pada program kami, untuk mencari node program ini menggunakan Binary Search yang membandingkan ID buku sebagai pembanding untuk mencari data yang dicari.

- i. searchBST() fungsi ini berguna untuk menyari ID Buku dengan mengambil elemen tengah lalu akan dicari ke kiri atau ke kanan.



```
● ● ● PerpustakaanLinkedList.c
// Fungsi untuk mencari node dalam BST
struct bstNode* searchBST(struct bstNode* root, int idBuku) {
    if (root == NULL || root->data.idBuku == idBuku)
        return root;

    if (idBuku < root->data.idBuku)
        return searchBST(root->left, idBuku);

    return searchBST(root->right, idBuku);
}
```

dengan hasilnya seperti ini

```
==== MENU BST ====
1. Lihat Semua Buku (Inorder)
2. Cari Buku
3. Tambah Buku
4. Hapus Buku
0. Home
Pilihan: 2
Masukkan ID Buku yang dicari: 13

==== BUKU DITEMUKAN ===

==== Buku 13 ===
Judul      : Partikel
Pengarang: Dee Lestari
Tahun      : 2012
Status     : Tersedia
=====
```

c. Menghapus leaf node, node yang memiliki 1 children dan 2 children

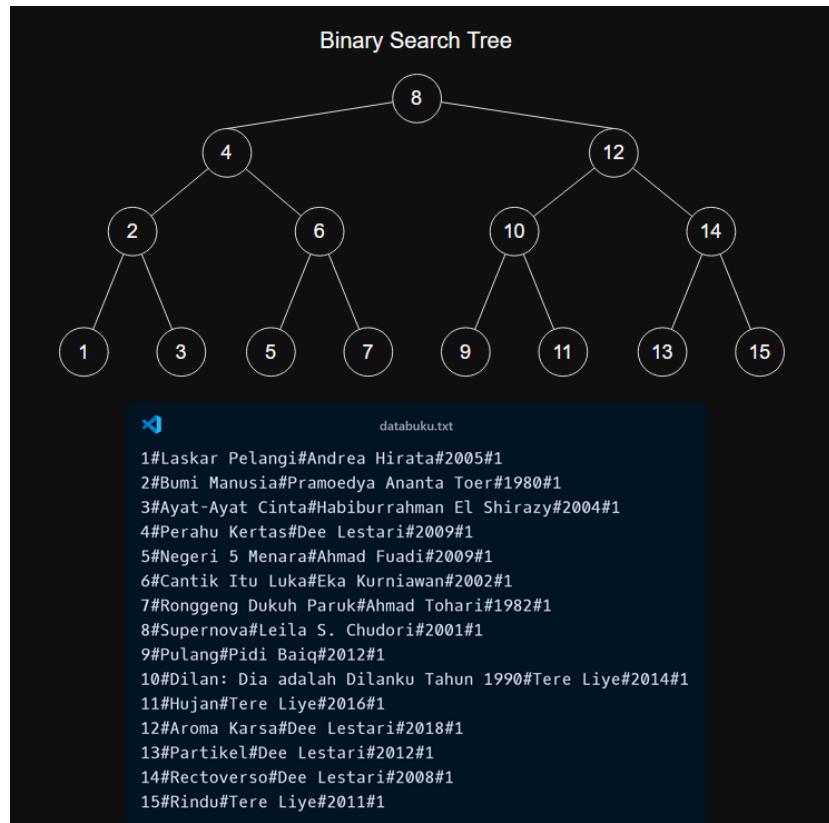
Pada program ini, kami hanya memakai 1 fungsi untuk mendelete node dengan menaruh beberapa kondisi. Fungsi akan mencari node mana yang akan di delete dan ketika sudah ketemu bakal masuk ke pengecekan kondisi lagi apakah mempunyai anak 1 atau anak 2 atau tidak memiliki anak sama sekali.

```
PerpustakaaperpustakaanW
// Fungsi untuk menghapus node dari BST
struct bstNode* deleteBST(struct bstNode* root, int idBuku) {
    if (root == NULL)
        return root;

    // Mencari node yang akan dihapus
    if (idBuku < root->data.idBuku)
        root->left = deleteBST(root->left, idBuku);
    else if (idBuku > root->data.idBuku)
        root->right = deleteBST(root->right, idBuku);
    else {
        // Node dengan satu atau tanpa child
        if (root->left == NULL) {
            struct bstNode* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL) {
            struct bstNode* temp = root->left;
            free(root);
            return temp;
        }

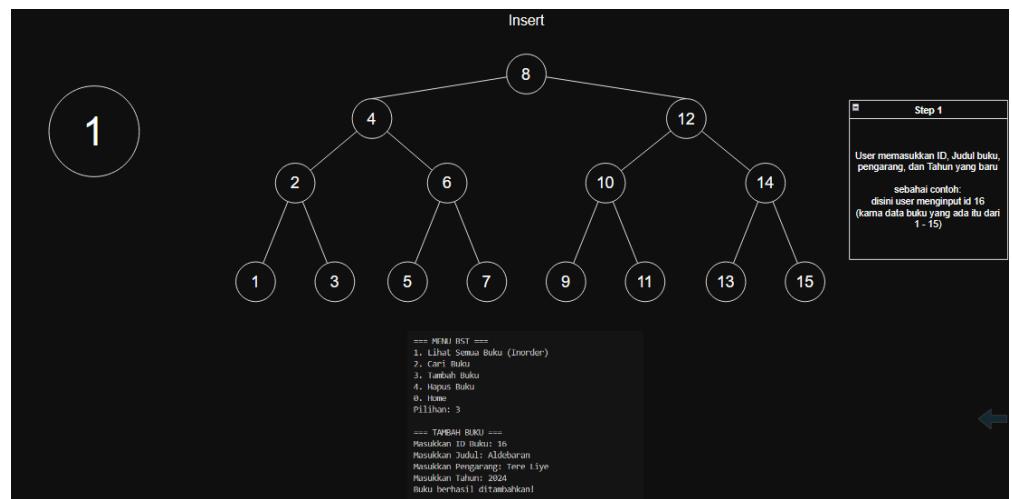
        // Node dengan dua children
        struct bstNode* temp = minValueNode(root->right);
        root->data = temp->data;
        root->right = deleteBST(root->right, temp->data.idBuku);
    }
    return root;
}
```

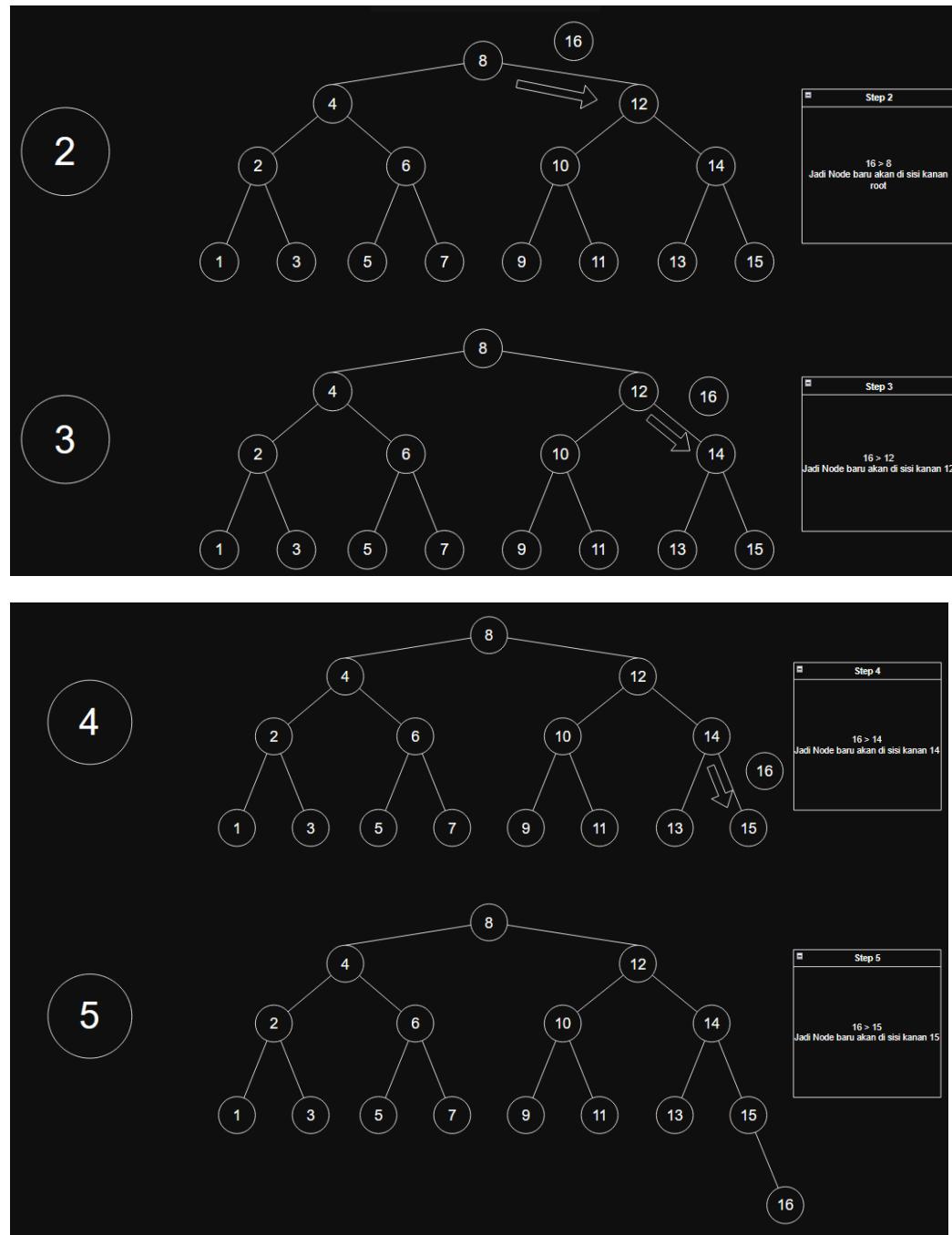
d. Bentuk dari Binary Search Tree



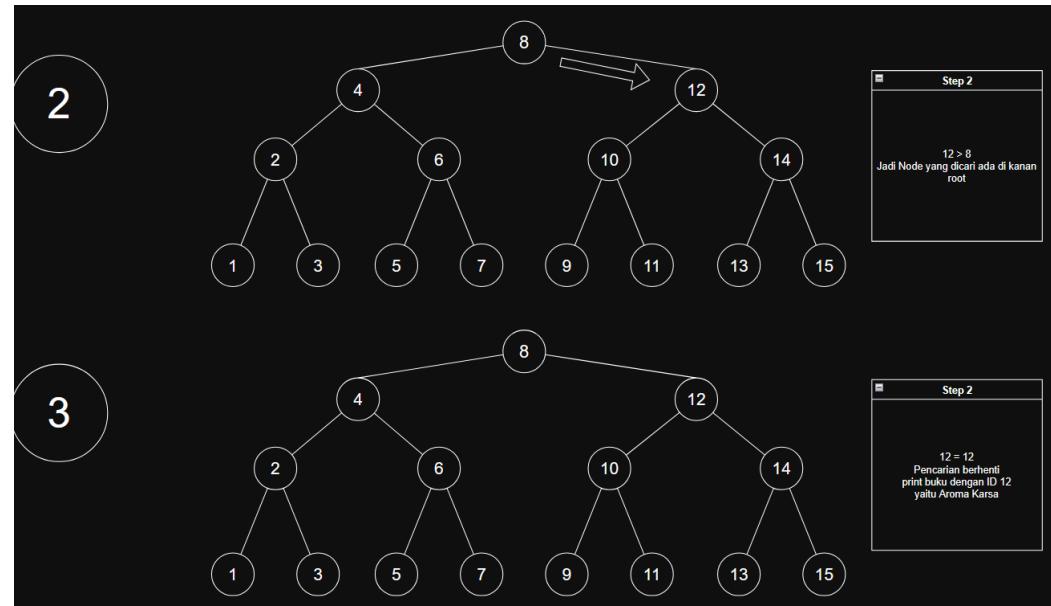
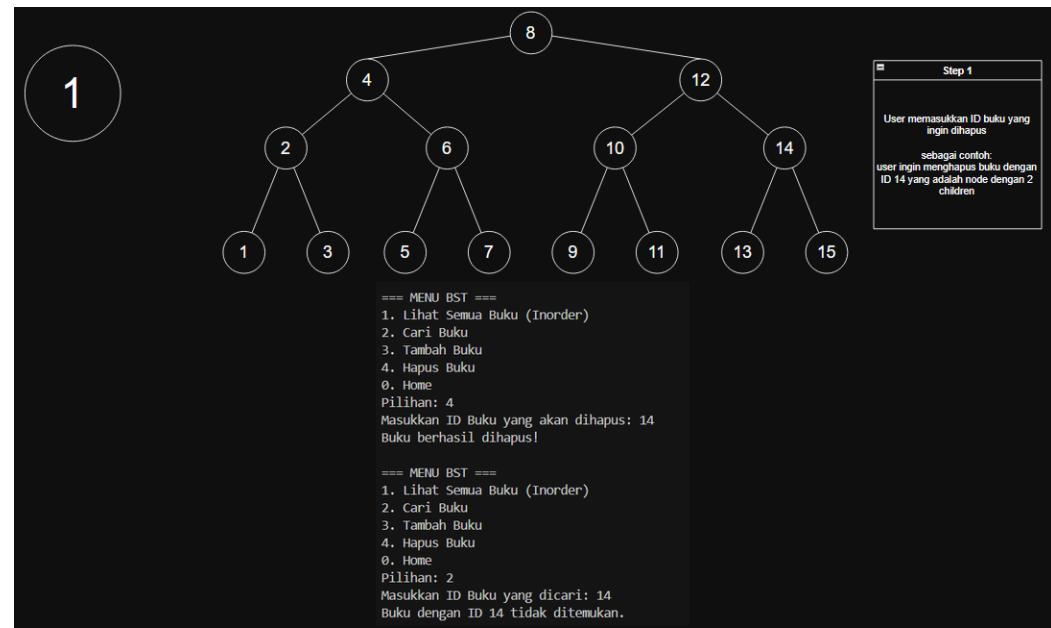
e. Alur Program berbentuk tree

i. Insert

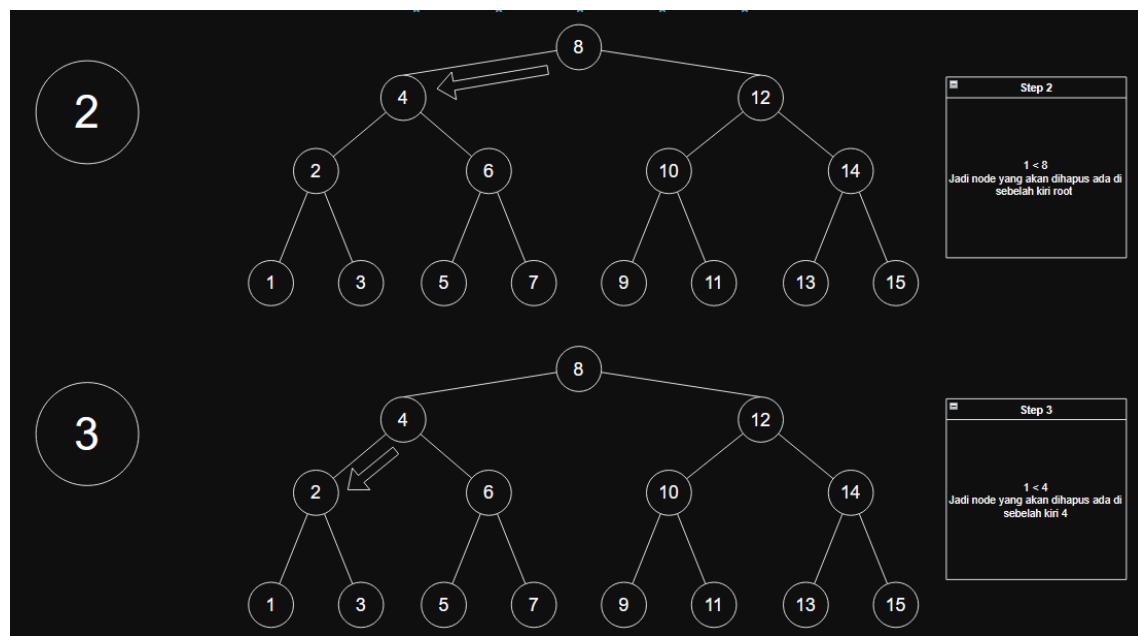
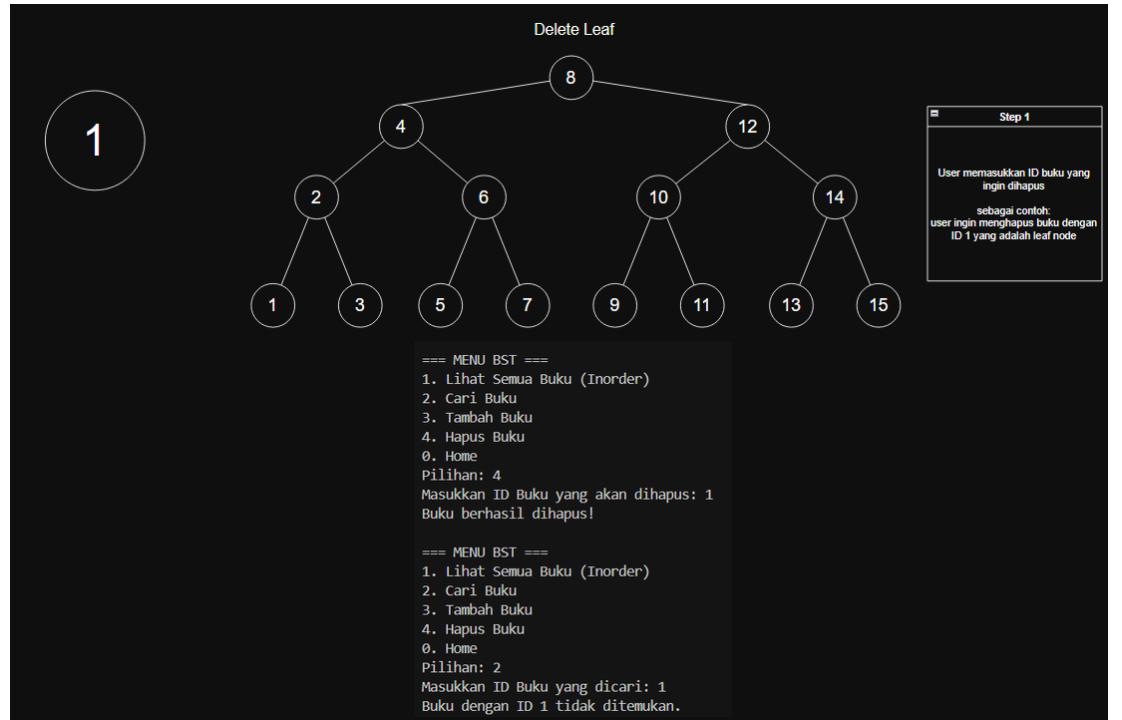


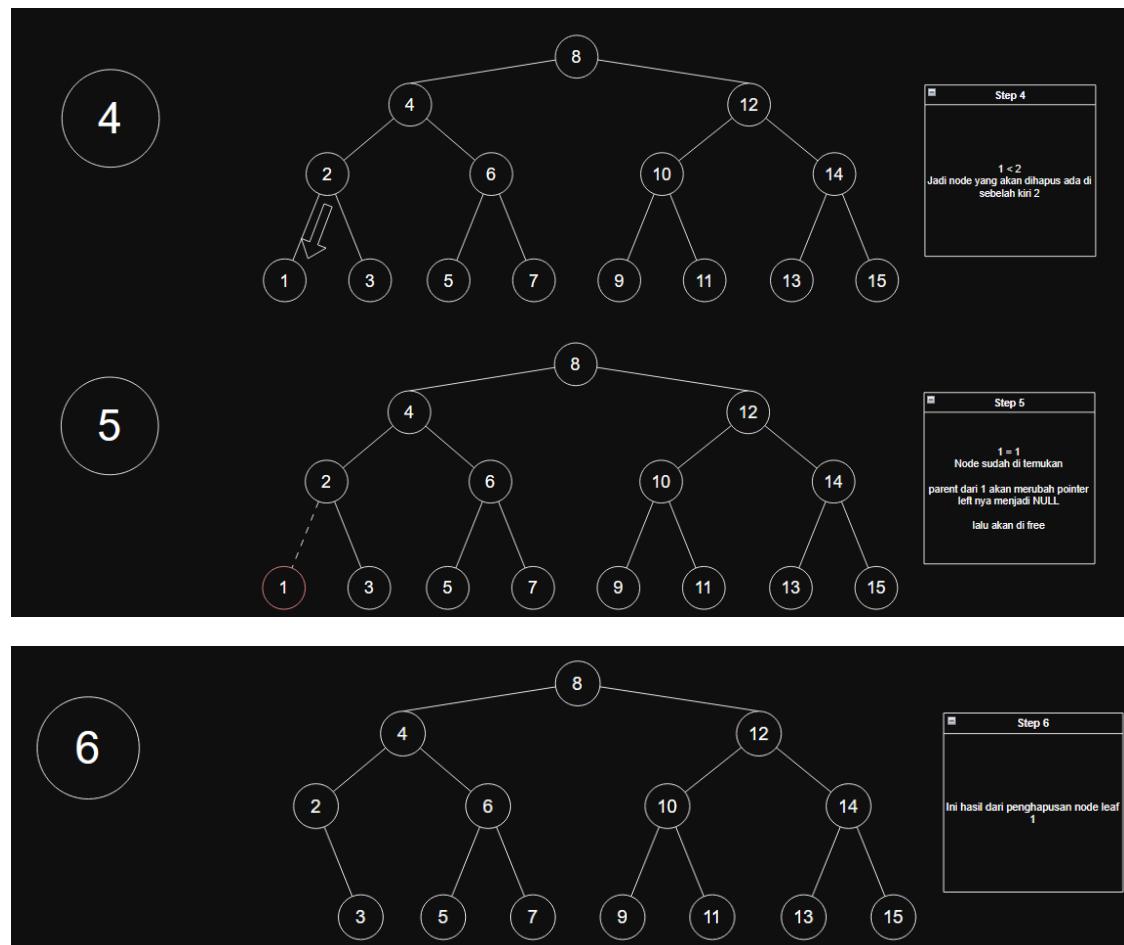


ii. Searching

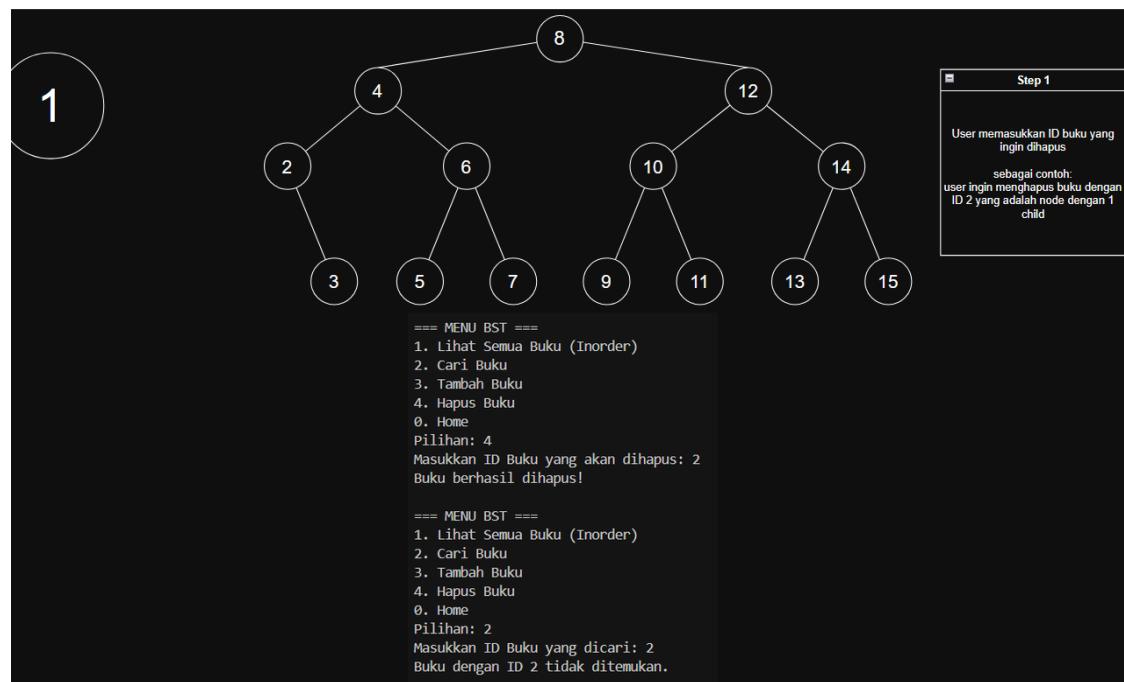


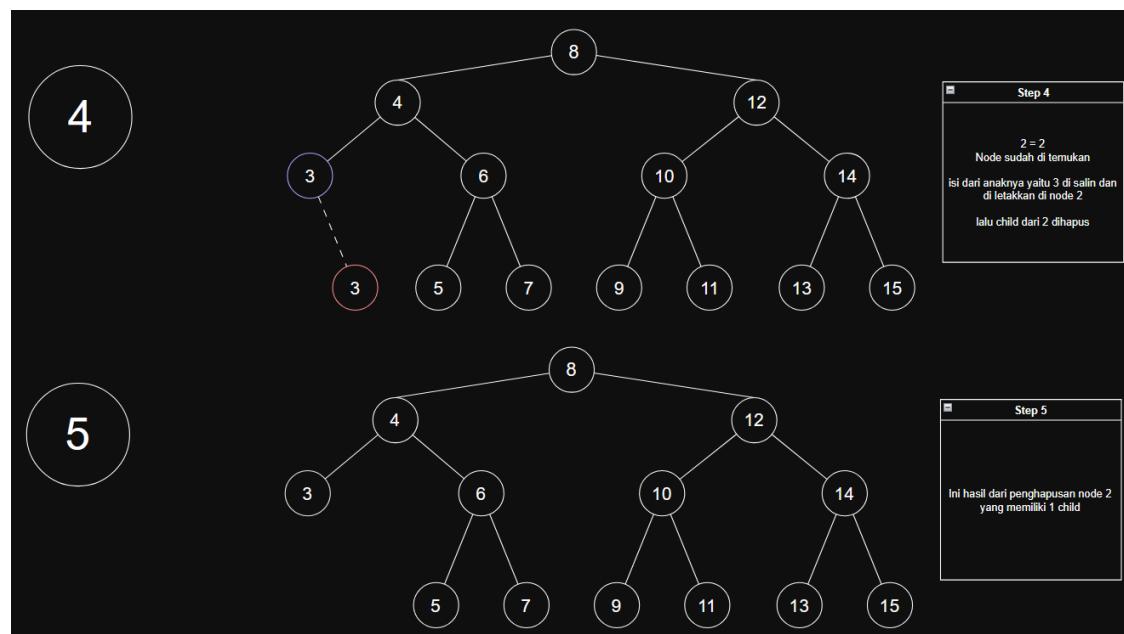
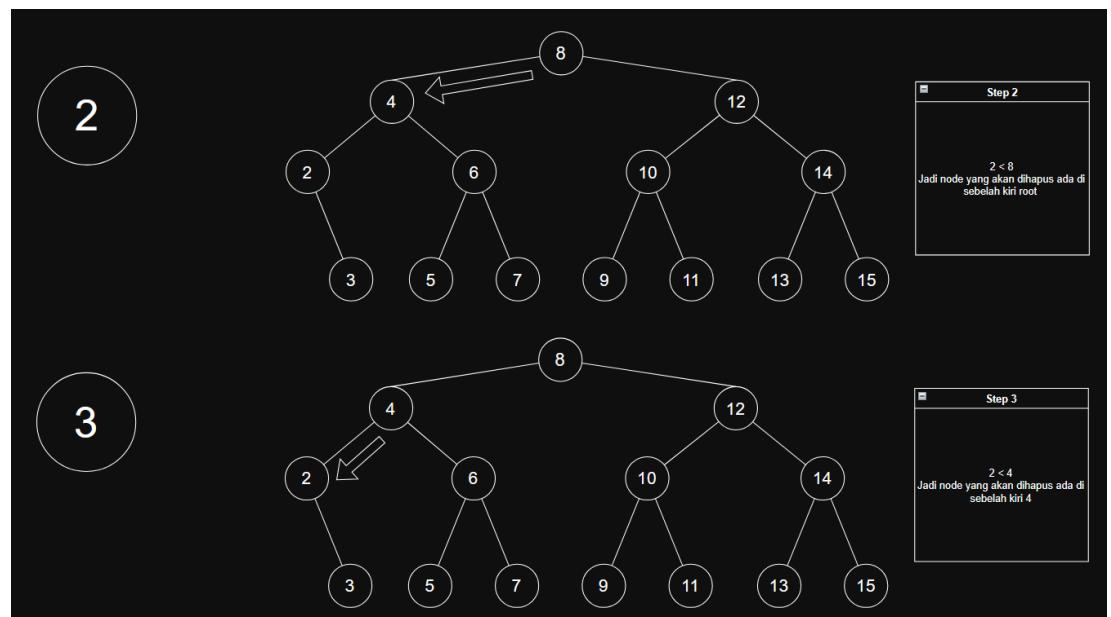
iii. Delete leaf node



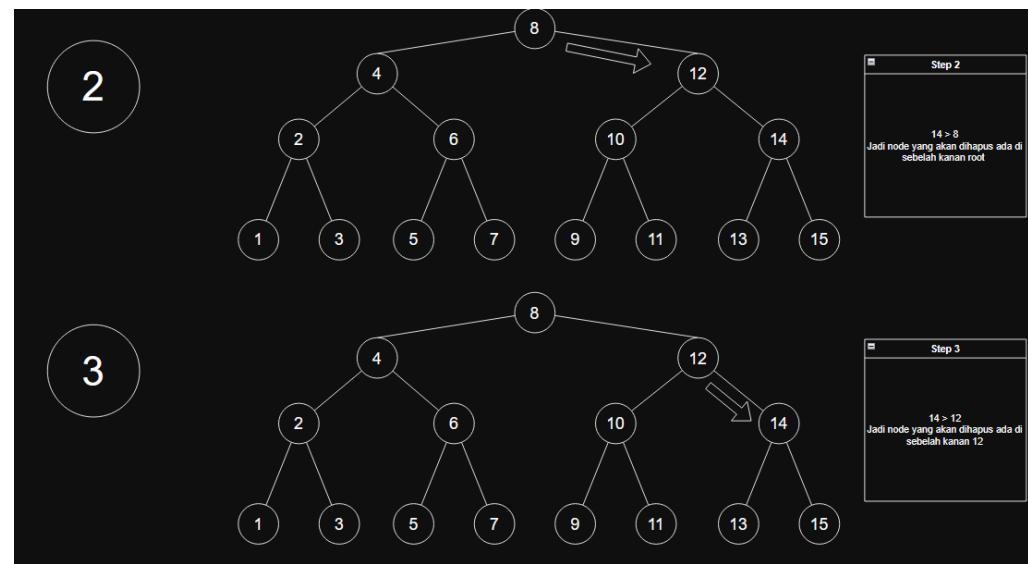
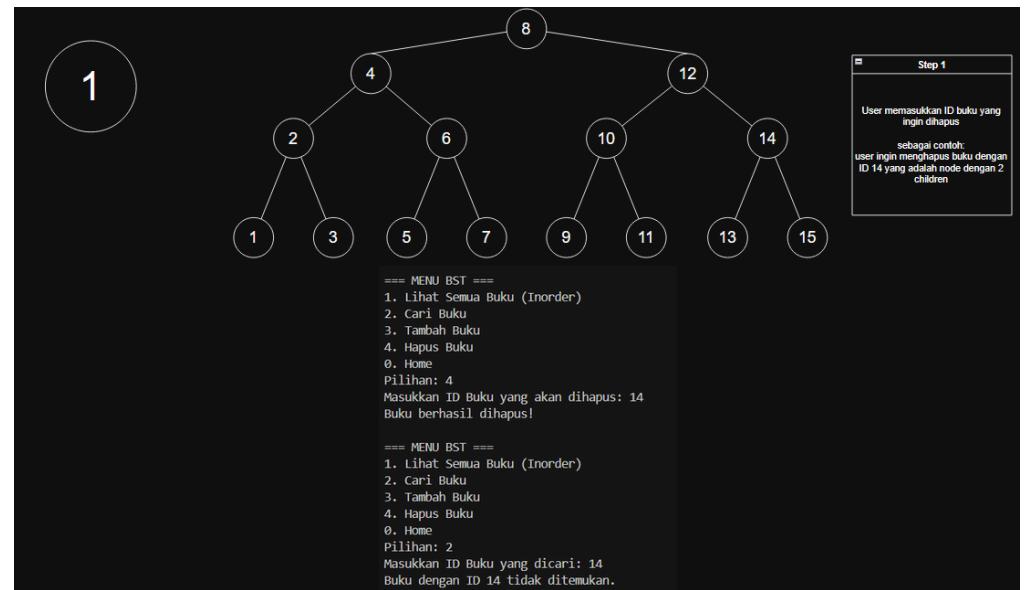


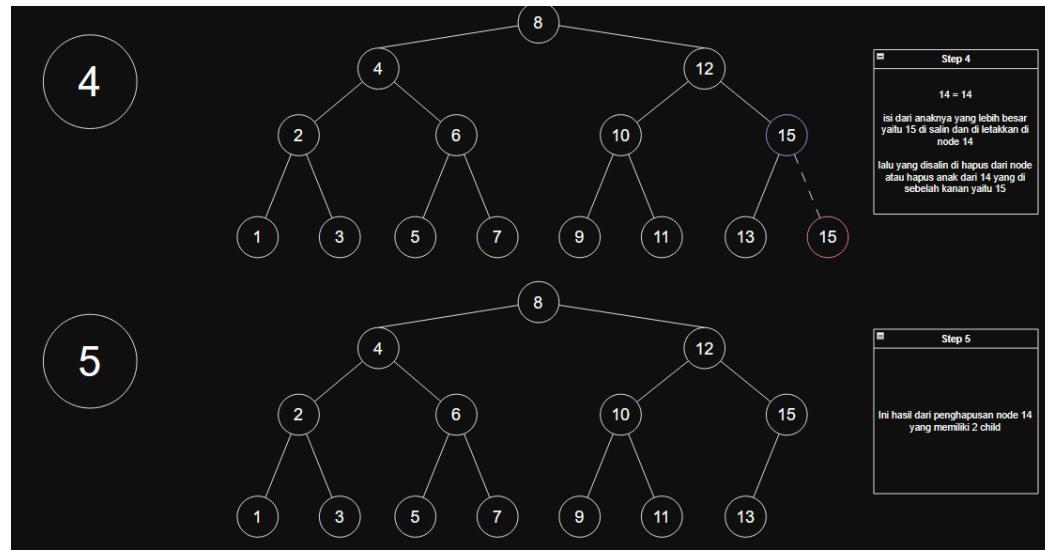
iv. Delete 1 Children





v. Delete 2 Children





2. Implementasi penggunaan Binary Heap

Selain menggunakan struktur data Binary Search Tree (BST), program ini juga mengimplementasikan Binary Heap untuk mempermudah pengelolaan prioritas buku. Program dapat menentukan buku mana yang memiliki prioritas lebih tinggi, misalnya berdasarkan ID Buku atau urutan alfabet judul buku, sehingga proses penambahan, penghapusan, dan pencarian prioritas dapat dilakukan dengan lebih efisien.

1. Membuat Binary Heap

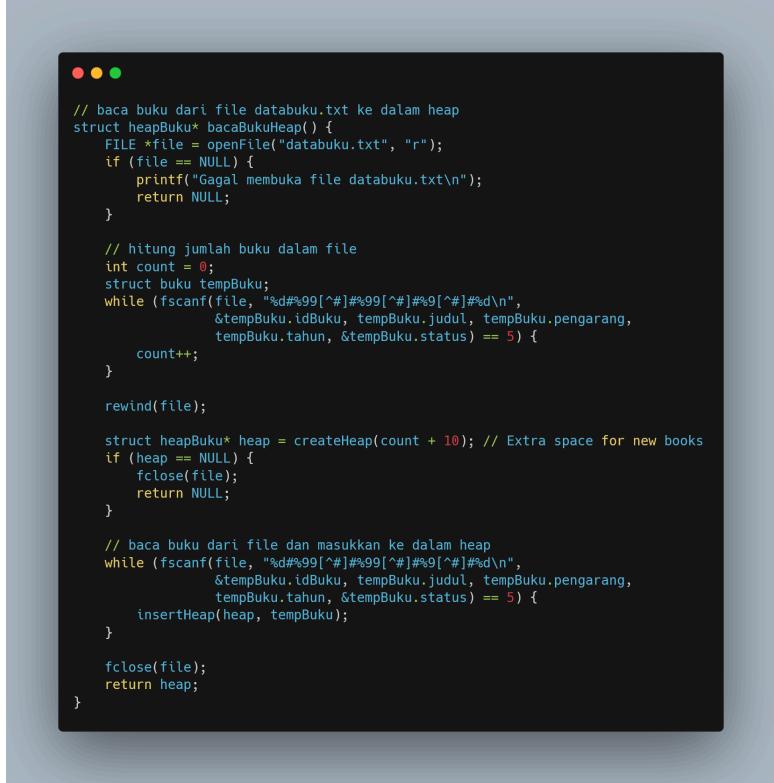
- i. `createHeap()` digunakan untuk membuat Heap kosong terlebih dahulu. Fungsi ini berguna untuk mengalokasikan memory agar sesuai dengan size dari datanya.



```
struct heapBuku* createHeap(int capacity) {
    struct heapBuku* heap = (struct heapBuku*)malloc(sizeof(struct heapBuku));
    if (heap == NULL) return NULL;

    heap->data = (struct buku*)malloc(capacity * sizeof(struct buku));
    if (heap->data == NULL) {
        free(heap);
        return NULL;
    }
    heap->size = 0;
    heap->capacity = capacity;
    return heap;
}
```

- ii. `bacaBukuHeap()` digunakan untuk membaca seluruh data yang ada di database `databuku.txt` lalu akan dimasukkan ke HEAP lalu dalam fungsi ini fungsi dari `createHeap()` akan dipanggil lagi untuk memasukan size dari data tersebut.



```
// baca buku dari file databuku.txt ke dalam heap
struct heapBuku* bacaBukuHeap() {
    FILE *file = openFile("databuku.txt", "r");
    if (file == NULL) {
        printf("Gagal membuka file databuku.txt\n");
        return NULL;
    }

    // hitung jumlah buku dalam file
    int count = 0;
    struct buku tempBuku;
    while (fscanf(file, "%d%99[^#]#%99[^#]#%9[^#]#%d\n",
                  &tempBuku.idBuku, tempBuku.judul, tempBuku.pengarang,
                  tempBuku.tahun, &tempBuku.status) == 5) {
        count++;
    }

    rewind(file);

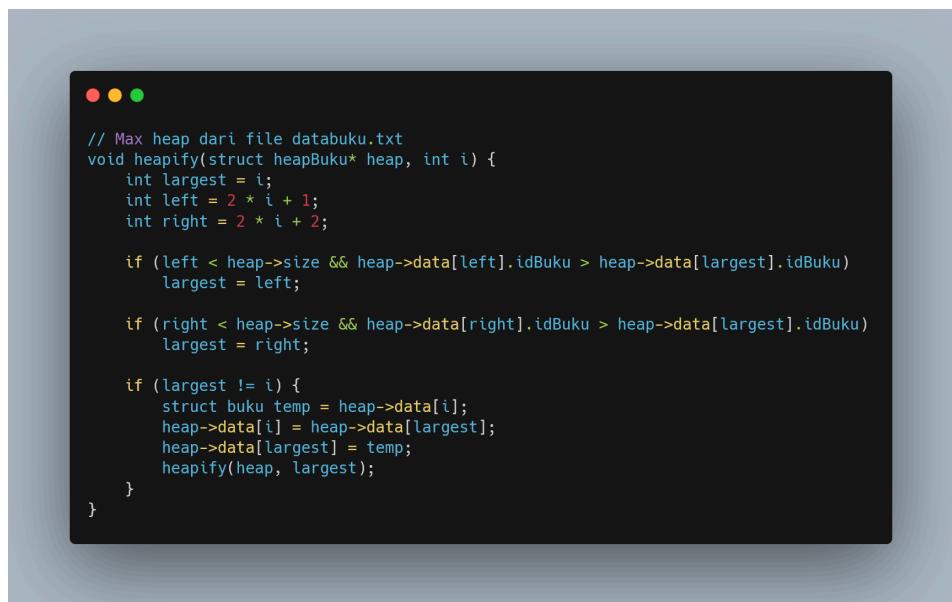
    struct heapBuku* heap = createHeap(count + 10); // Extra space for new books
    if (heap == NULL) {
        fclose(file);
        return NULL;
    }

    // baca buku dari file dan masukkan ke dalam heap
    while (fscanf(file, "%d%99[^#]#%99[^#]#%9[^#]#%d\n",
                  &tempBuku.idBuku, tempBuku.judul, tempBuku.pengarang,
                  tempBuku.tahun, &tempBuku.status) == 5) {
        insertHeap(heap, tempBuku);
    }

    fclose(file);
    return heap;
}
```

2. Konversi ke Heapify

- buildHeap() akan digunakan untuk membuat pilihan heapify dari mau yang dari terkecil(MinHeap) atau terbesar(MaxHeap) berdasarkan pilihan User.



```
// Max heap dari file databuku.txt
void heapify(struct heapBuku* heap, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && heap->data[left].idBuku > heap->data[largest].idBuku)
        largest = left;

    if (right < heap->size && heap->data[right].idBuku > heap->data[largest].idBuku)
        largest = right;

    if (largest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[largest];
        heap->data[largest] = temp;
        heapify(heap, largest);
    }
}
```

- ii. Heapify() akan membuat Heap menjadi MaxHeap berdasarkan besar ID buku. jadi setiap parent akan memiliki ID buku yang lebih besar ketimbang Childnya.



```
// Max heap dari file databuku.txt
void heapify(struct heapBuku* heap, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && heap->data[left].idBuku > heap->data[largest].idBuku)
        largest = left;

    if (right < heap->size && heap->data[right].idBuku > heap->data[largest].idBuku)
        largest = right;

    if (largest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[largest];
        heap->data[largest] = temp;
        heapify(heap, largest);
    }
}
```

Contoh dari hasilnya

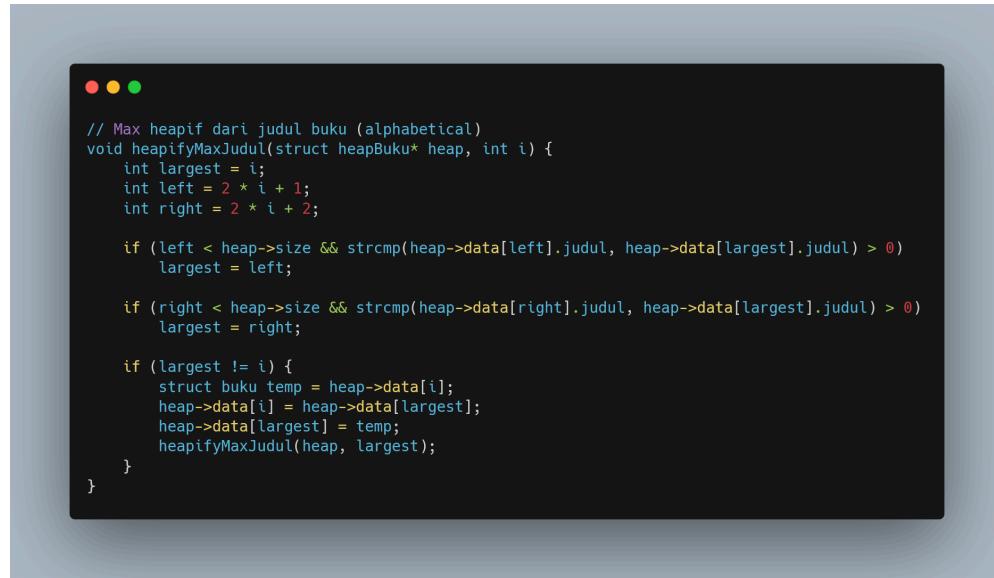
```
    === Buku 1 dari 15 ===

    === Buku 15 ===
Judul      : Rindu
Pengarang: Tere Liye
Tahun      : 2011
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
    === Buku 2 dari 15 ===

    === Buku 11 ===
Judul      : Hujan
Pengarang: Tere Liye
Tahun      : 2016
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
    === Buku 3 dari 15 ===

    === Buku 14 ===
Judul      : Rectoverso
Pengarang: Dee Lestari
Tahun      : 2008
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

- iii. HeapifyMaxJudul() sama seperti Heapify() namun ini menggunakan judul buku. Huruf akan menjadi pertimbangan apakah lebih besar atau lebih kecil.



```
// Max heapif dari judul buku (alphabetical)
void heapifyMaxJudul(struct heapBuku* heap, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && strcmp(heap->data[left].judul, heap->data[largest].judul) > 0)
        largest = left;

    if (right < heap->size && strcmp(heap->data[right].judul, heap->data[largest].judul) > 0)
        largest = right;

    if (largest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[largest];
        heap->data[largest] = temp;
        heapifyMaxJudul(heap, largest);
    }
}
```

Contoh hasilnya

```
== Buku 1 dari 15 ==

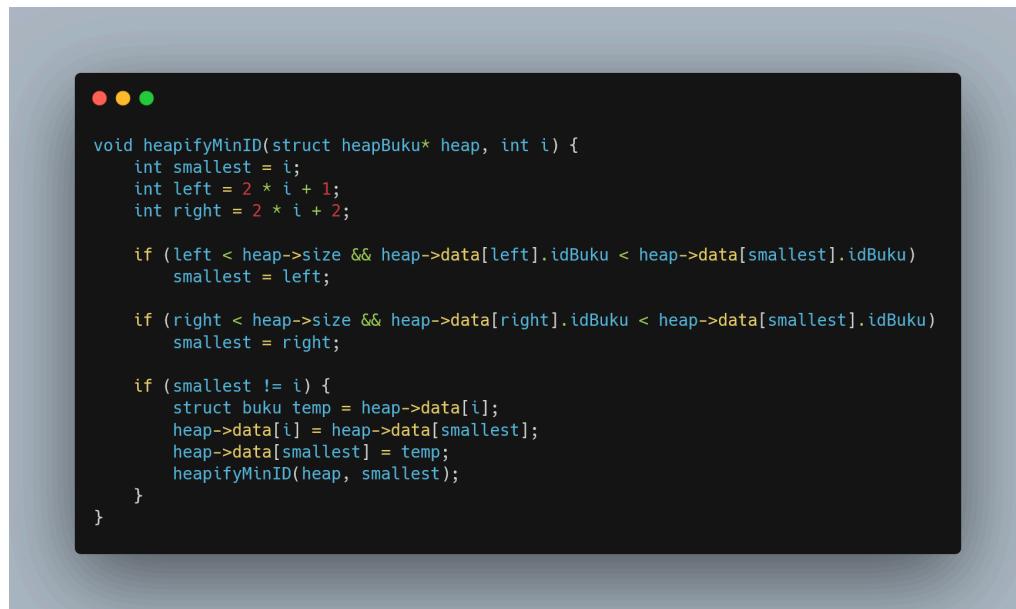
== Buku 8 ==
Judul      : Supernova
Pengarang: Leila S. Chudori
Tahun      : 2001
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 2 dari 15 ==

== Buku 9 ==
Judul      : Pulang
Pengarang: Pidi Baiq
Tahun      : 2012
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 3 dari 15 ==

== Buku 7 ==
Judul      : Ronggeng Dukuh Paruk
Pengarang: Ahmad Tohari
Tahun      : 1982
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 4 dari 15 ==

== Buku 4 ==
Judul      : Perahu Kertas
Pengarang: Dee Lestari
Tahun      : 2009
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: nn
```

- iv. HeapifyMinID() ini berfungsi untuk membuat MinHeap berdasarkan ID buku, setiap parentnya akan memiliki ID buku yang lebih kecil dari kedua childnya



```
void heapifyMinID(struct heapBuku* heap, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && heap->data[left].idBuku < heap->data[smallest].idBuku)
        smallest = left;

    if (right < heap->size && heap->data[right].idBuku < heap->data[smallest].idBuku)
        smallest = right;

    if (smallest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[smallest];
        heap->data[smallest] = temp;
        heapifyMinID(heap, smallest);
    }
}
```

contoh dari hasilnya

```
==== Buku 1 dari 15 ===

==== Buku 1 ===
Judul      : Laskar Pelangi
Pengarang: Andrea Hirata
Tahun      : 2005
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
==== Buku 2 dari 15 ===

==== Buku 2 ===
Judul      : Bumi Manusia
Pengarang: Pramoedya Ananta Toer
Tahun      : 1980
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
==== Buku 3 dari 15 ===

==== Buku 3 ===
Judul      : Ayat-Ayat Cinta
Pengarang: Habiburrahman El Shirazy
Tahun      : 2004
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
==== Buku 4 dari 15 ===

==== Buku 4 ===
Judul      : Perahu Kertas
Pengarang: Dee Lestari
Tahun      : 2009
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

- v. HeapifyMinJudul() akan membuat MinHeap berdasarkan huruf yang ada di judul.

```
● ● ●

// Min heapif dari judul buku (alphabetical)
void heapifyMinJudul(struct heapBuku* heap, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && strcmp(heap->data[left].judul, heap->data[smallest].judul) < 0)
        smallest = left;

    if (right < heap->size && strcmp(heap->data[right].judul, heap->data[smallest].judul) < 0)
        smallest = right;

    if (smallest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[smallest];
        heap->data[smallest] = temp;
        heapifyMinJudul(heap, smallest);
    }
}
```

Contoh dari hasilnya

```
==== Buku 1 dari 15 ===

==== Buku 12 ===
Judul      : Aroma Karsa
Pengarang: Dee Lestari
Tahun      : 2018
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: nn
==== Buku 2 dari 15 ===

==== Buku 2 ===
Judul      : Bumi Manusia
Pengarang: Pramoedya Ananta Toer
Tahun      : 1980
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: nn
==== Buku 3 dari 15 ===

==== Buku 3 ===
Judul      : Ayat-Ayat Cinta
Pengarang: Habiburrahman El Shirazy
Tahun      : 2004
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
==== Buku 4 dari 15 ===

==== Buku 4 ===
Judul      : Perahu Kertas
Pengarang: Dee Lestari
Tahun      : 2009
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

3. Menghapus elemen yang ada di binary Heap

- i. HapusBukuPrioritas() ini adalah fungsi untuk memanggil ExtractMax() agar bisa menghapus 1 elemen paling besar.



```
// Fungsi untuk menambah buku ke heap
void tambahBukuHeap(int idUser, char username[], struct heapBuku* heap, int heapType, int sortBy) {
    if (heap == NULL) {
        printf("Heap belum dibuat. Silakan buat heap terlebih dahulu.\n");
        printf("\nTekan Enter untuk melanjutkan...");
        getchar();
        return;
    }

    struct buku bukuBaru;
    printf("\n==== TAMBAH BUKU KE HEAP ====\n");
    printf("ID Buku: ");
    if (scanf("%d", &bukuBaru.idBuku) != 1) {
        while (getchar() != '\n');
        printf("ID tidak valid!\n");
        return;
    }
    while (getchar() != '\n');

    printf("Judul: ");
    scanf(" %99[^\\n]", bukuBaru.judul);

    printf("Pengarang: ");
    scanf(" %99[^\\n]", bukuBaru.pengarang);

    printf("Tahun: ");
    scanf(" %9[^\\n]", bukuBaru.tahun);

    bukuBaru.status = 1; // Default tersedia

    insertHeap(heap, bukuBaru);
    // Rebuild heap sesuai konfigurasi yang dipilih user
    buildHeap(heap, heapType, sortBy);
    printf("Buku berhasil ditambahkan ke heap!\n");

    printf("\nTekan Enter untuk melanjutkan...");
    getchar();
}
```

- ii. ExtractMax() ini adalah fungsi yang kami pakai untuk menghapus 1 elemen di Heap dan kami mengprioritaskan yang tertinggi untuk dihapus terlebih dahulu.

```
● ● ●

// ekstrak buku dengan prioritas tertinggi dari heap
struct buku extractMax(struct heapBuku* heap) {
    struct buku emptyBook = {0};
    if (heap->size <= 0) {
        return emptyBook;
    }

    struct buku max = heap->data[0];
    heap->data[0] = heap->data[heap->size - 1];
    heap->size--;

    if (heap->size > 0) {
        heapify(heap, 0); // Default ke max heap berdasarkan ID
    }

    return max;
}
```

Contoh penggunaannya seperti ini

```
==== BUKU DENGAN PRIORITAS TERTINGGI DIHAPUS ====
Heap Type: MIN HEAP
Kriteria: Judul Buku
=====
==== Buku 12 ===
Judul      : Aroma Karsa
Pengarang: Dee Lestari
Tahun      : 2018
Status     : Tersedia
=====
Buku berhasil dihapus dari heap!

Tekan Enter untuk melanjutkan...
```

4.

Insertion data ke Heap

- i. tambahBukuHeap() berfungsi untuk user bisa memasukkan buku yang ingin ditambahkan dengan memasukkan id, judul, pengarang, dan tahun. Status pada buku sudah default 1 atau tersedia.



```
● ● ●

// Fungsi untuk menambah buku ke heap
void tambahBukuHeap(int idUser, char username[], struct heapBuku* heap, int heapType, int sortBy) {
    if (heap == NULL) {
        printf("Heap belum dibuat. Silakan buat heap terlebih dahulu.\n");
        printf("\nTekan Enter untuk melanjutkan...");
        getchar();
        return;
    }

    struct buku bukuBaru;
    printf("\n--- TAMBAH BUKU KE HEAP ---\n");
    printf("ID Buku: ");
    if (scanf("%d", &bukuBaru.idBuku) != 1) {
        while (getchar() != '\n');
        printf("ID tidak valid!\n");
        return;
    }
    while (getchar() != '\n');

    printf("Judul: ");
    scanf(" %99[^\\n]", bukuBaru.judul);

    printf("Pengarang: ");
    scanf(" %99[^\\n]", bukuBaru.pengarang);

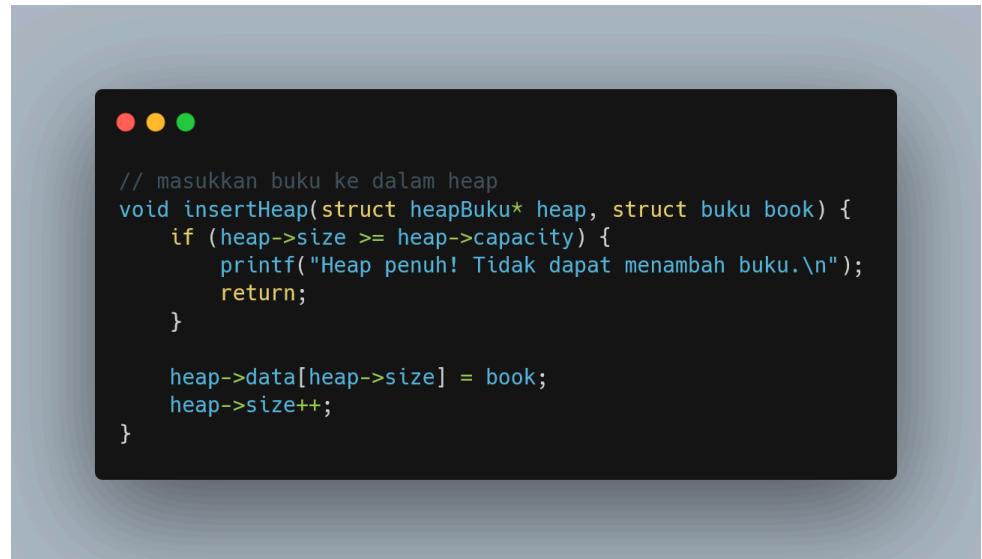
    printf("Tahun: ");
    scanf(" %9[\\n]", bukuBaru.tahun);

    bukuBaru.status = 1; // Default tersedia

    insertHeap(heap, bukuBaru);
    // Rebuild heap sesuai konfigurasi yang dipilih user
    buildHeap(heap, heapType, sortBy);
    printf("Buku berhasil ditambahkan ke heap!\n");

    printf("\nTekan Enter untuk melanjutkan...");
    getchar();
}
```

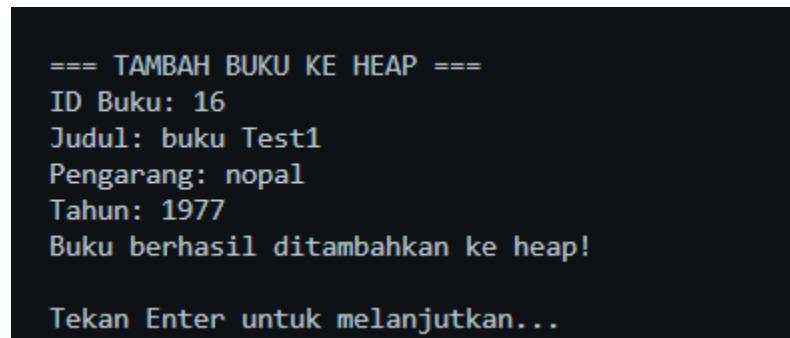
- ii. InsertHeap() untuk memasukkan data yang sudah diinput oleh user dari fungsi tambahBukuHeap() ke Heap



```
// masukkan buku ke dalam heap
void insertHeap(struct heapBuku* heap, struct buku book) {
    if (heap->size >= heap->capacity) {
        printf("Heap penuh! Tidak dapat menambah buku.\n");
        return;
    }

    heap->data[heap->size] = book;
    heap->size++;
}
```

hasilnya bakal seperti ini



```
==== TAMBAH BUKU KE HEAP ====
ID Buku: 16
Judul: buku Test1
Pengarang: nopal
Tahun: 1977
Buku berhasil ditambahkan ke heap!

Tekan Enter untuk melanjutkan...
```

saat ditampilkan

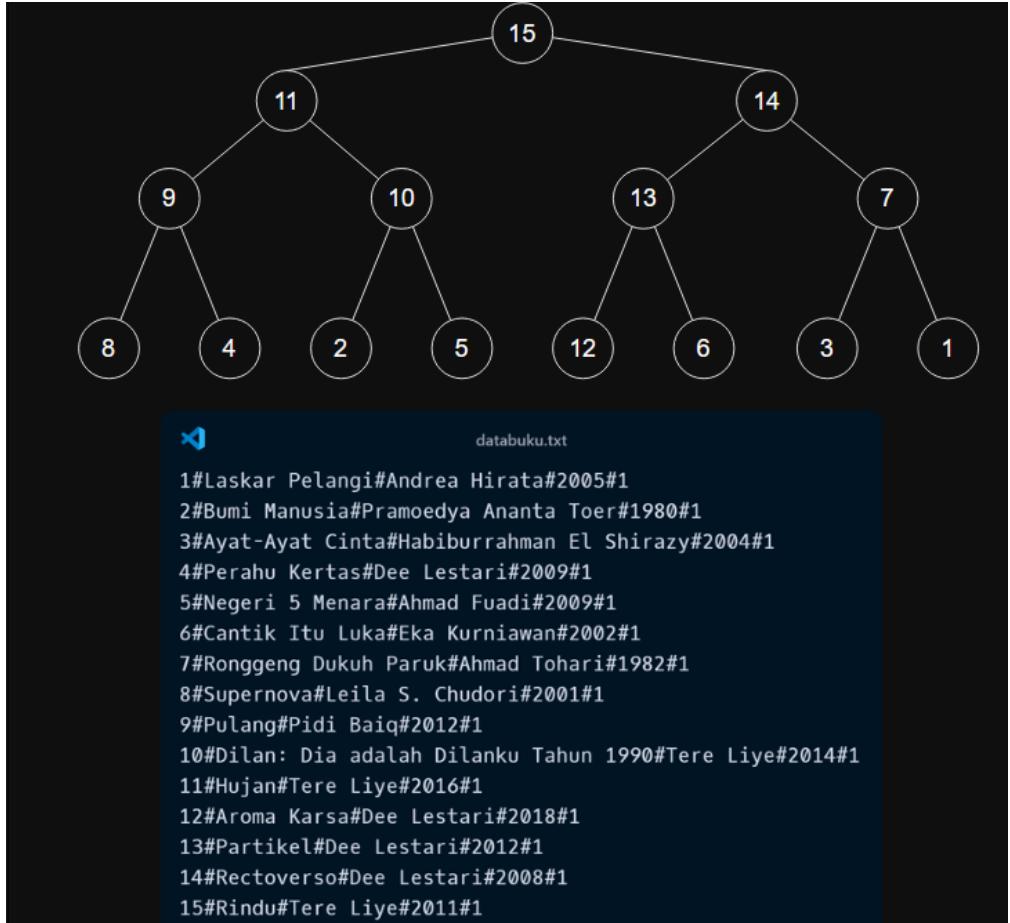


```
==== Buku 1 dari 16 ===

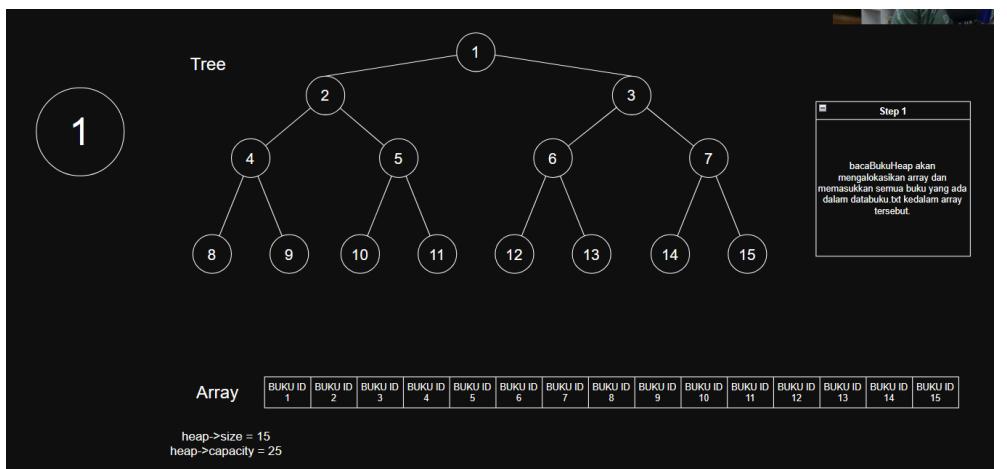
==== Buku 16 ===
Judul      : buku Test1
Pengarang: nopal
Tahun      : 1977
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: █
```

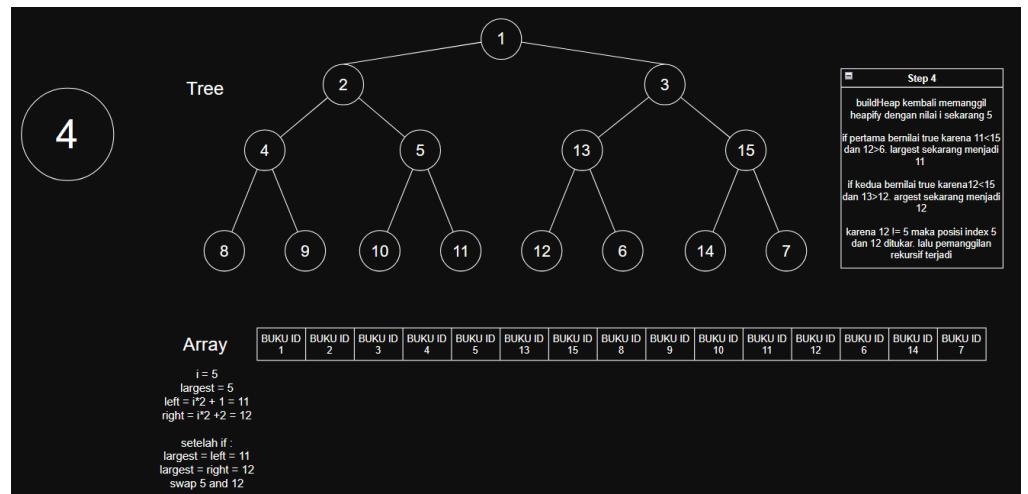
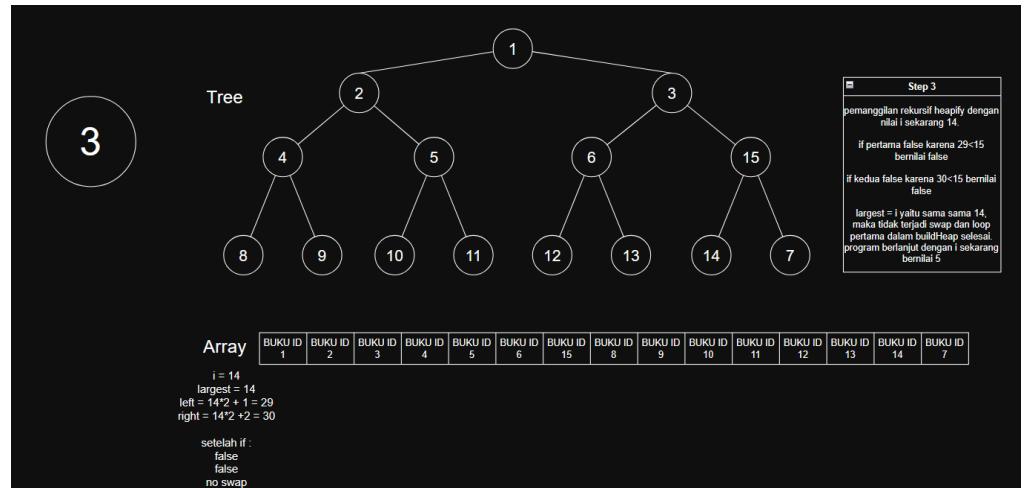
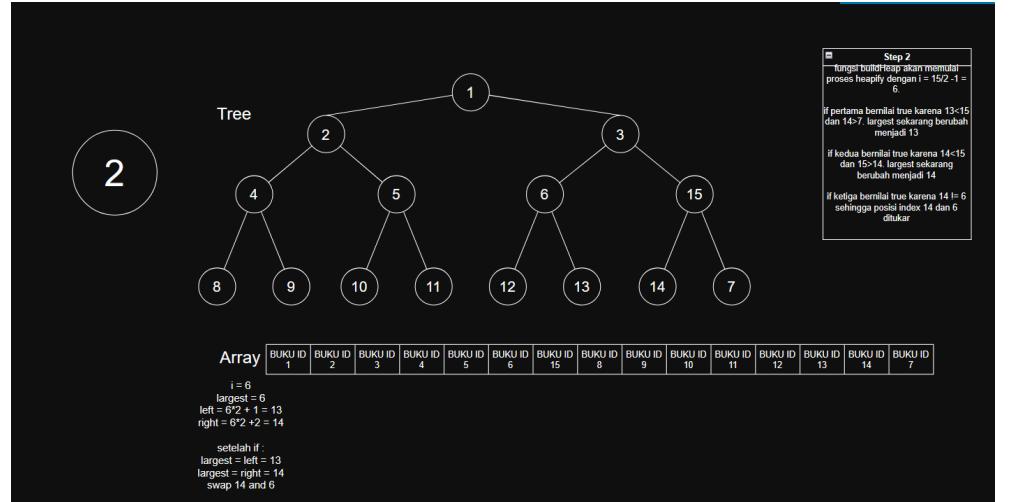
3. Alur program menggunakan Binary Heap Tree

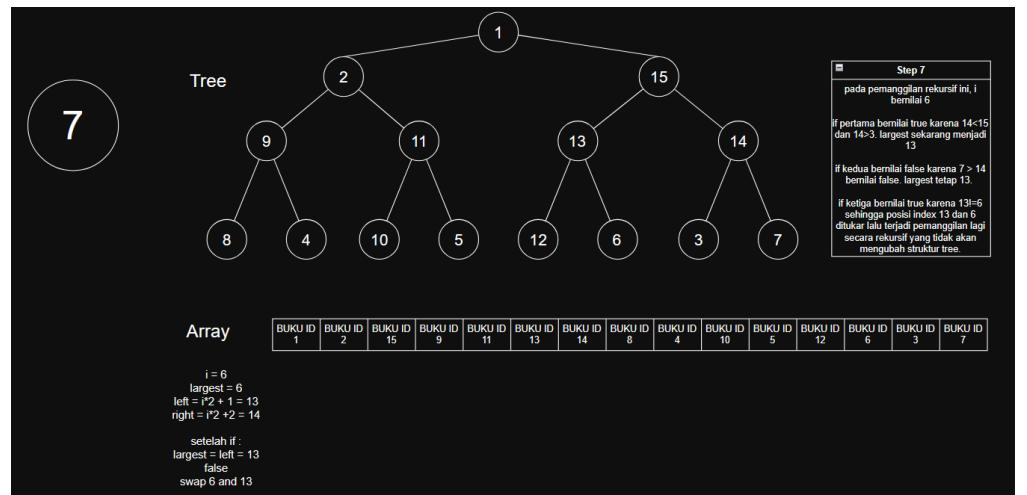
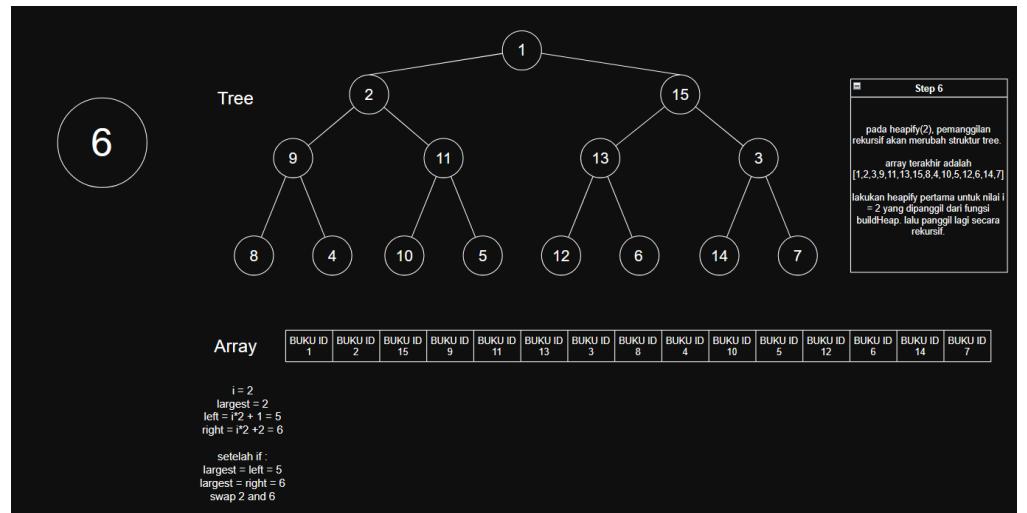
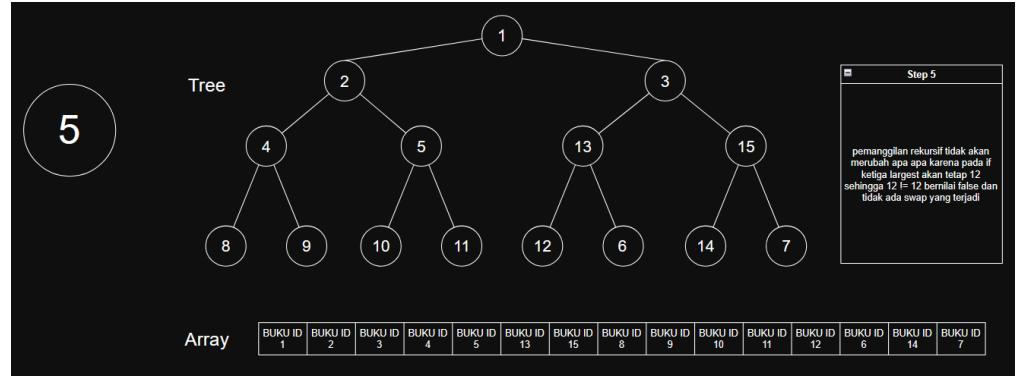
i. Bentuk dari MaxHeap Tree

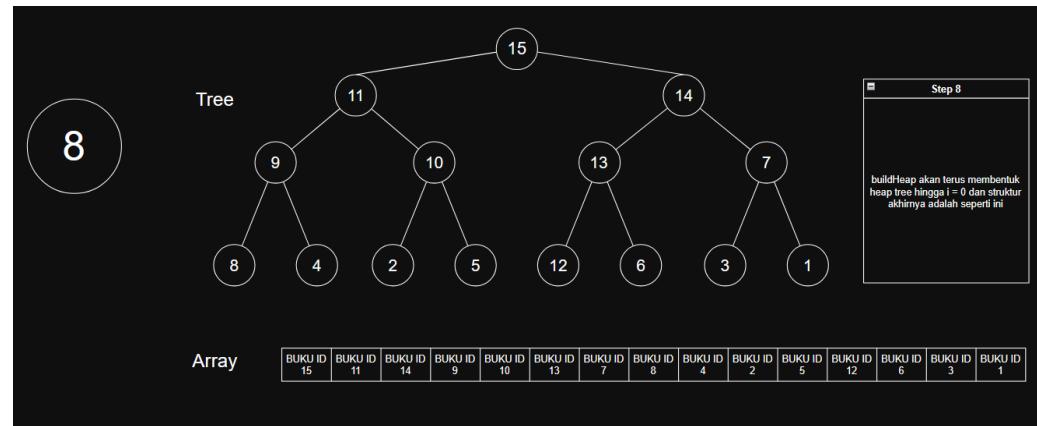


ii. Pembuatan MaxHeap Tree

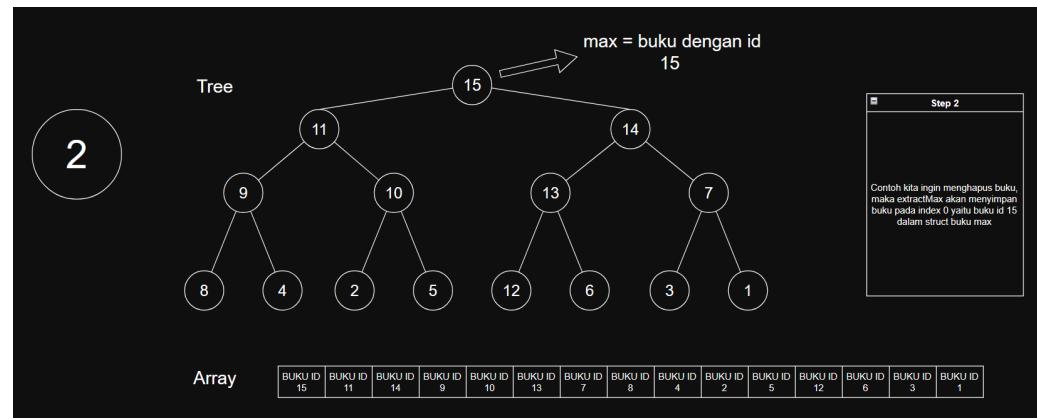
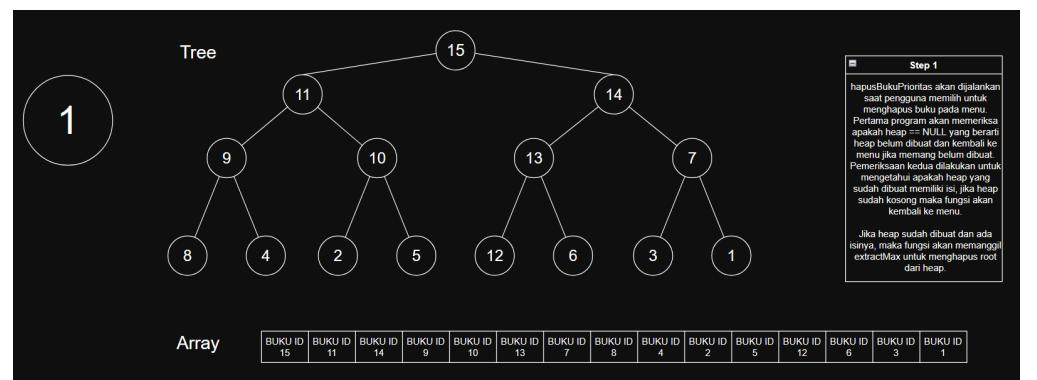


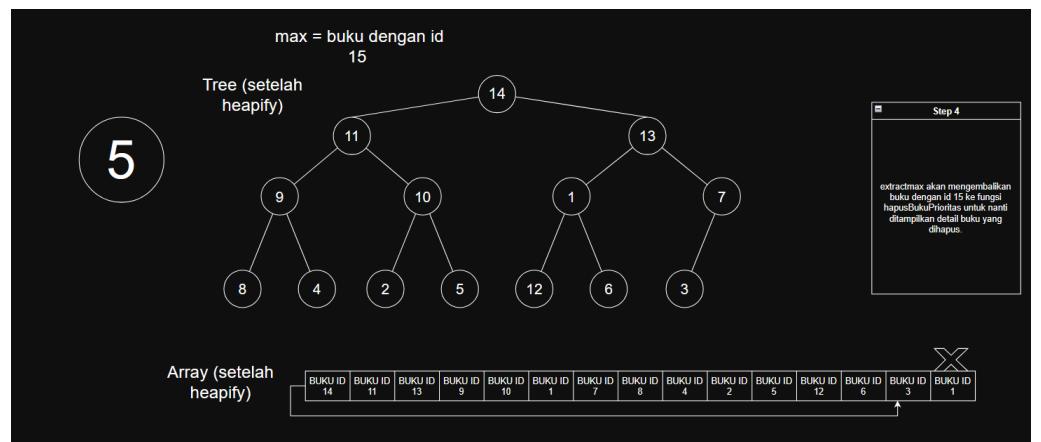
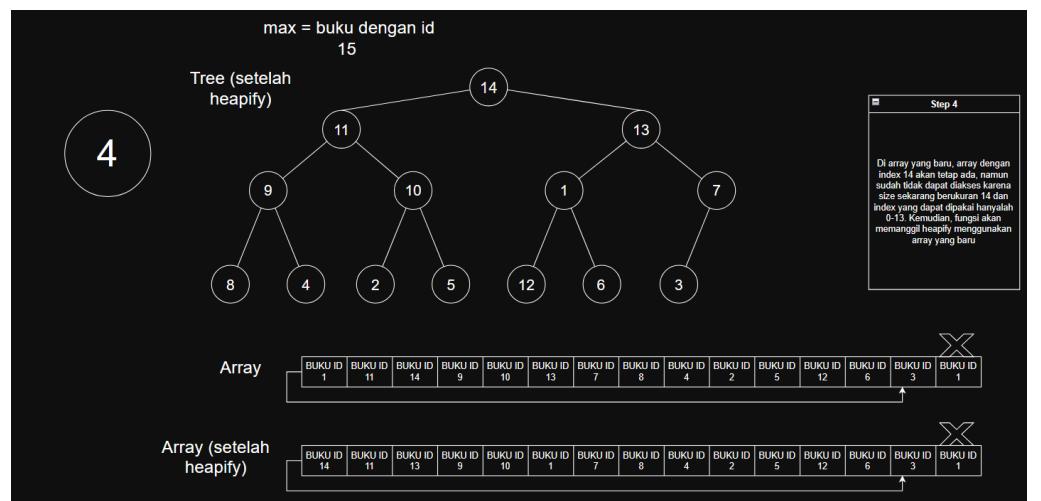
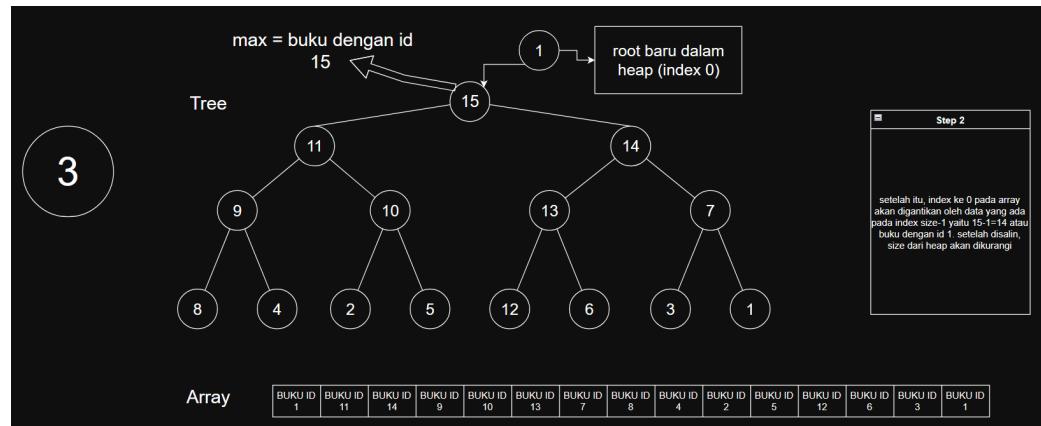






iii. Delete Node di MaxHeap Tree





3. Implementasi Algoritma Sorting

a. Heap Sort

seperti pada Implementasi Binairy Heap, kami menggunakan heap Sort dengan memanfaatkan heapify kami bisa mengsorting datanya dari MaxHeap dan

MinHeap. Kami juga memberi pilihan apakah ini di sorting secara MinHeap atau MaxHeap lalu juga apakah mau secara judul buku atau ID Buku.

- i. heapify() akan membuat Heap menjadi MaxHeap berdasarkan besar ID buku. jadi setiap parent akan memiliki ID buku yang lebih besar ketimbang Childnya.



```
// Max heap dari file databuku.txt
void heapify(struct heapBuku* heap, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && heap->data[left].idBuku > heap->data[largest].idBuku)
        largest = left;

    if (right < heap->size && heap->data[right].idBuku > heap->data[largest].idBuku)
        largest = right;

    if (largest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[largest];
        heap->data[largest] = temp;
        heapify(heap, largest);
    }
}
```

- ii. heapifyMaxJudul() sama seperti Heapify() namun ini menggunakan judul buku. Huruf akan menjadi pertimbangan apakah lebih besar atau lebih kecil.

```
// Max heapif dari judul buku (alphabetical)
void heapifyMaxJudul(struct heapBuku* heap, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && strcmp(heap->data[left].judul, heap->data[largest].judul) > 0)
        largest = left;

    if (right < heap->size && strcmp(heap->data[right].judul, heap->data[largest].judul) > 0)
        largest = right;

    if (largest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[largest];
        heap->data[largest] = temp;
        heapifyMaxJudul(heap, largest);
    }
}
```

- iii. `heapifyMinID()` ini berfungsi untuk membuat MinHeap berdasarkan ID buku, setiap parentnya akan memiliki ID buku yang lebih kecil dari kedua childnya.

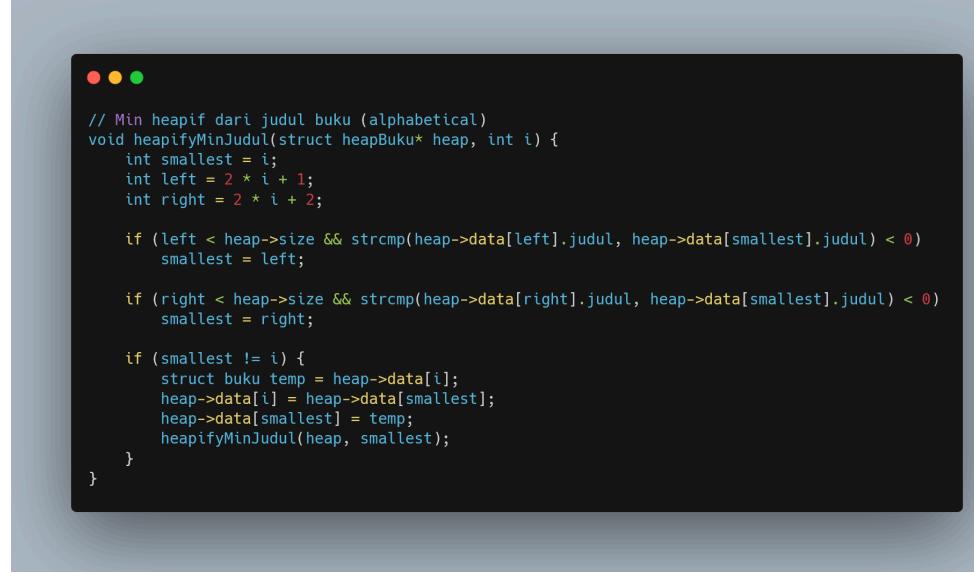
```
void heapifyMinID(struct heapBuku* heap, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && heap->data[left].idBuku < heap->data[smallest].idBuku)
        smallest = left;

    if (right < heap->size && heap->data[right].idBuku < heap->data[smallest].idBuku)
        smallest = right;

    if (smallest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[smallest];
        heap->data[smallest] = temp;
        heapifyMinID(heap, smallest);
    }
}
```

- iv. `heapifyMinJudul()` mirip seperti `HeapifyMinID()`, bedanya ini menggunakan huruf alfabet dari judul buku untuk menjadi pembandingnya.



```
// Min heapif dari judul buku (alphabetical)
void heapifyMinJudul(struct heapBuku* heap, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < heap->size && strcmp(heap->data[left].judul, heap->data[smallest].judul) < 0)
        smallest = left;

    if (right < heap->size && strcmp(heap->data[right].judul, heap->data[smallest].judul) < 0)
        smallest = right;

    if (smallest != i) {
        struct buku temp = heap->data[i];
        heap->data[i] = heap->data[smallest];
        heap->data[smallest] = temp;
        heapifyMinJudul(heap, smallest);
    }
}
```

v. Hasil dari Heap Sorting

1. MinHeap menggunakan ID

```
== Buku 1 dari 15 ==
== Buku 1 ==
Judul      : Laskar Pelangi
Pengarang: Andrea Hirata
Tahun      : 2005
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 2 dari 15 ==

== Buku 2 ==
Judul      : Bumi Manusia
Pengarang: Pramoedya Ananta Toer
Tahun      : 1980
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 3 dari 15 ==

== Buku 3 ==
Judul      : Ayat-Ayat Cinta
Pengarang: Habiburrahman El Shirazy
Tahun      : 2004
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
== Buku 4 dari 15 ==

== Buku 4 ==
Judul      : Perahu Kertas
Pengarang: Dee Lestari
Tahun      : 2009
Status     : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

2. MinHeap menggunakan Judul

```
    === Buku 12 ===
    Judul      : Aroma Karsa
    Pengarang: Dee Lestari
    Tahun      : 2018
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
    === Buku 2 dari 15 ===

    === Buku 2 ===
    Judul      : Bumi Manusia
    Pengarang: Pramoedya Ananta Toer
    Tahun      : 1980
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
    === Buku 3 dari 15 ===

    === Buku 3 ===
    Judul      : Ayat-Ayat Cinta
    Pengarang: Habiburrahman El Shirazy
    Tahun      : 2004
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
    === Buku 4 dari 15 ===

    === Buku 4 ===
    Judul      : Perahu Kertas
    Pengarang: Dee Lestari
    Tahun      : 2009
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
```

3. MaxHeap menggunakan ID

```
--- Buku 15 ---
Judul    : Rindu
Pengarang: Tere Liye
Tahun    : 2011
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 2 dari 15 ---

--- Buku 11 ---
Judul    : Hujan
Pengarang: Tere Liye
Tahun    : 2016
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 3 dari 15 ---

--- Buku 14 ---
Judul    : Rectoverso
Pengarang: Dee Lestari
Tahun    : 2008
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 4 dari 15 ---

--- Buku 9 ---
Judul    : Pulang
Pengarang: Pidi Baiq
Tahun    : 2012
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

4. MaxHeap menggunakan Judul

```
--- Buku 8 ---
Judul    : Supernova
Pengarang: Leila S. Chudori
Tahun    : 2001
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 2 dari 15 ---

--- Buku 9 ---
Judul    : Pulang
Pengarang: Pidi Baiq
Tahun    : 2012
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 3 dari 15 ---

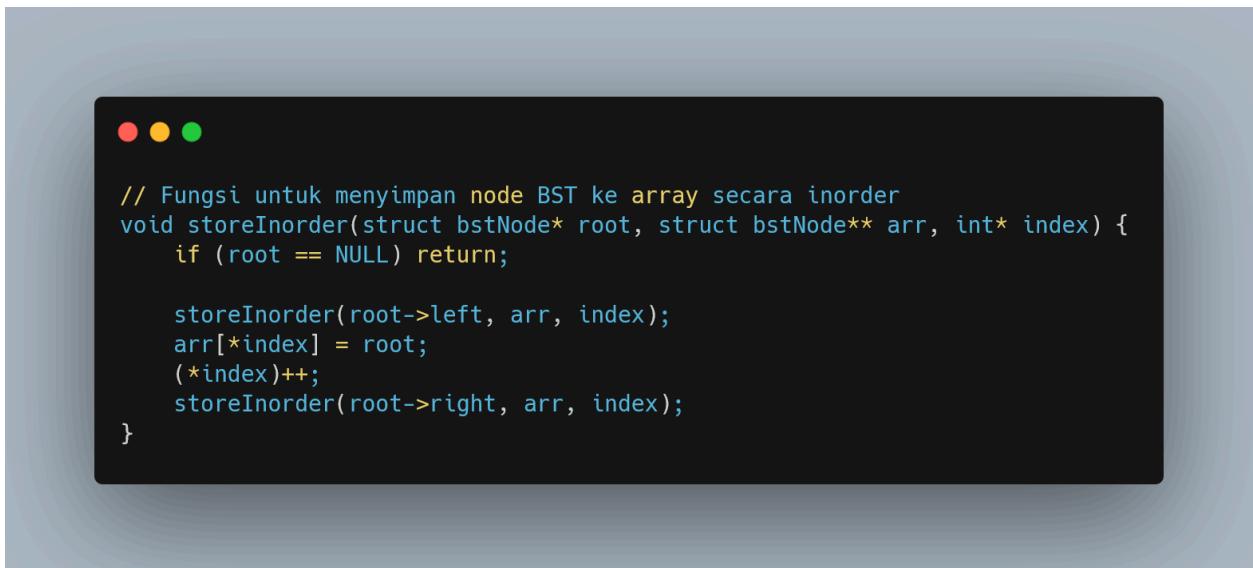
--- Buku 7 ---
Judul    : Ronggeng Dukuh Paruk
Pengarang: Ahmad Tohari
Tahun    : 1982
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
--- Buku 4 dari 15 ---

--- Buku 4 ---
Judul    : Perahu Kertas
Pengarang: Dee Lestari
Tahun    : 2009
Status   : Tersedia
=====
Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
Pilihan: n
```

b. Quick Sort

Quick sort mengurutkan array hasil BST berdasarkan huruf dari Judul buku atau ID Buku. Kami memberi pilihan cara sortingnya dari mau menggunakan ID saja atau berdasarkan Judul buku secara A ke Z atau Z ke A.

- i. `storeInOrder()` fungsi ini akan mengurutkan node yang ada di BST ke array untuk sebagai inorder. Fungsi ini juga berguna sebagai mengsorting apabila ingin memakai ID buku.



```
// Fungsi untuk menyimpan node BST ke array secara inorder
void storeInorder(struct bstNode* root, struct bstNode** arr, int* index) {
    if (root == NULL) return;

    storeInorder(root->left, arr, index);
    arr[*index] = root;
    (*index)++;
    storeInorder(root->right, arr, index);
}
```

- ii. `lihatBukuUrutan()` sebagai quick sort, berfungsi untuk mengsorting secara huruf alfabet dari A-Z atau Z-A.



```
void lihatBukuUrutan(struct bstNode** nodes, int count, int sortType) {
    if (sortType == 1) { // untuk A-Z
        qsort(nodes, count, sizeof(struct bstNode*), compareJudulAZ);
    } else if (sortType == 2) { // untuk Z-A
        qsort(nodes, count, sizeof(struct bstNode*), compareJudulZA);
    }

    int currentIndex = 0;
    char pilhan;

    while (1) {
        tampilPerBuku(&(nodes[currentIndex]->data));

        printf("Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali\n");
        printf("Pilihan: ");
        scanf(" %c", &pilhan);

        switch (pilhan) {
            case 'n':
                if (currentIndex < count - 1) {
                    currentIndex++;
                } else {
                    currentIndex = 0;
                }
                break;
            case 'p':
                if (currentIndex > 0) {
                    currentIndex--;
                } else {
                    currentIndex = count - 1;
                }
                break;
            case 'q':
                return;
            default:
                printf("Pilihan tidak valid!\n");
        }
    }
}
```

- iii. compareJudulAZ dan compareJudulZA untuk mengsorting judul buku berdasarkan huruf depannya.



```
// Fungsi untuk membandingkan judul buku (A-Z)
int compareJudulAZ(const void* a, const void* b) {
    struct bstNode* nodeA = *(struct bstNode***)a;
    struct bstNode* nodeB = *(struct bstNode***)b;
    return strcmp(nodeA->data.judul, nodeB->data.judul);
}

// Fungsi untuk membandingkan judul buku (Z-A)
int compareJudulZA(const void* a, const void* b) {
    struct bstNode* nodeA = *(struct bstNode***)a;
    struct bstNode* nodeB = *(struct bstNode***)b;
    return strcmp(nodeB->data.judul, nodeA->data.judul);
}
```

- iv. Hasil Sorting dari Quick Sort seperti ini

1. Menggunakan ID

```
    === Buku 1 ===
    Judul      : Laskar Pelangi
    Pengarang: Andrea Hirata
    Tahun      : 2005
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 2 ===
    Judul      : Bumi Manusia
    Pengarang: Pramoedya Ananta Toer
    Tahun      : 1980
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 3 ===
    Judul      : Ayat-Ayat Cinta
    Pengarang: Habiburrahman El Shirazy
    Tahun      : 2004
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 4 ===
    Judul      : Perahu Kertas
    Pengarang: Dee Lestari
    Tahun      : 2009
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
```

2. Menggunakan Judul Buku A-Z

```
    === Buku 12 ===
    Judul      : Aroma Karsa
    Pengarang: Dee Lestari
    Tahun      : 2018
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 3 ===
    Judul      : Ayat-Ayat Cinta
    Pengarang: Habiburrahman El Shirazy
    Tahun      : 2004
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 2 ===
    Judul      : Bumi Manusia
    Pengarang: Pramoedya Ananta Toer
    Tahun      : 1980
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 6 ===
    Judul      : Cantik Itu Luka
    Pengarang: Eka Kurniawan
    Tahun      : 2002
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
```

3. Menggunakan Judul Buku Z-A

```
    === Buku 8 ===
    Judul      : Supernova
    Pengarang: Leila S. Chudori
    Tahun      : 2001
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 7 ===
    Judul      : Ronggeng Dukuh Paruk
    Pengarang: Ahmad Tohari
    Tahun      : 1982
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

    === Buku 15 ===
    Judul      : Rindu
    Pengarang: Tere Liye
    Tahun      : 2011
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n

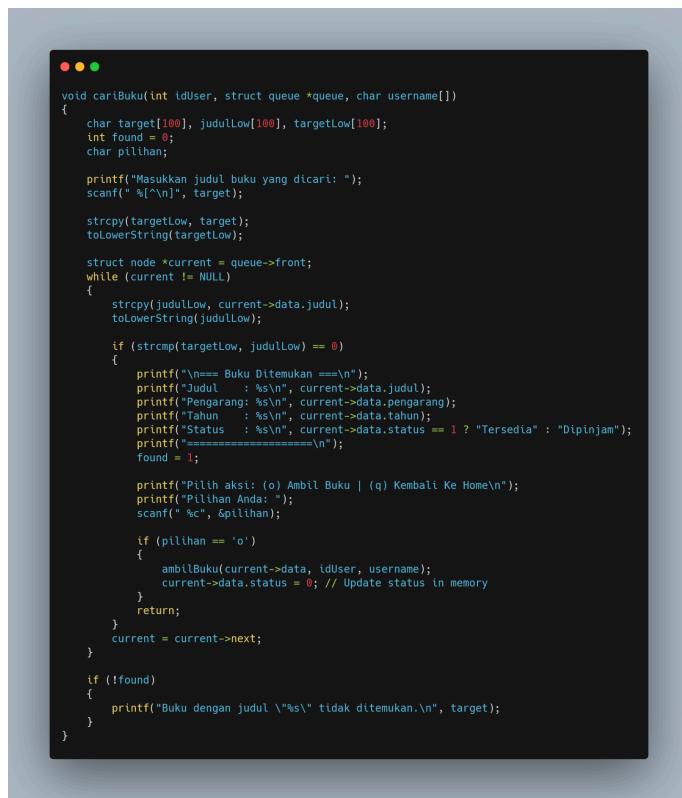
    === Buku 14 ===
    Judul      : Rectoverso
    Pengarang: Dee Lestari
    Tahun      : 2008
    Status     : Tersedia
    =====
    Pilih aksi: (n) Buku Selanjutnya | (p) Buku Sebelumnya | (q) Kembali
    Pilihan: n
```

4. Implementasi Algoritma Searching

a. Linear Search

Linear Search adalah pencarian sederhana di mana setiap elemen dalam struktur data diperiksa satu per satu hingga ditemukan data yang dicari atau hingga seluruh elemen telah diperiksa. Pada program ini, kami memakai linear search pada pilihan Linear dan juga Heap.

- i. cariBuku() pada fungsi ini akan jalan ketika user memilih Linear. Fungsi ini akan mencari buku yang di input oleh user lalu program akan mengubah inputan user jadi huruf kecil dan akan dibandingkan.



```
void cariBuku(int idUser, struct queue *queue, char username[])
{
    char target[100], judulLow[100], targetLow[100];
    int found = 0;
    char pilihan;

    printf("Masukkan judul buku yang dicari: ");
    scanf(" %[^\n]", target);

    strcpy(targetLow, target);
    toLowerString(targetLow);

    struct node *current = queue->front;
    while (current != NULL)
    {
        strcpy(judulLow, current->data.judul);
        toLowerString(judulLow);

        if (strcmp(targetLow, judulLow) == 0)
        {
            printf("\n==== Buku Ditemukan ====\n");
            printf("Judul : $s\n", current->data.judul);
            printf("Pengarang: $s\n", current->data.pengarang);
            printf("Tahun : $s\n", current->data.tahun);
            printf("Status : $s\n", current->data.status == 1 ? "Tersedia" : "Dipinjam");
            printf("=====\\n ");

            if (pilihan == 'o')
            {
                ambilBuku(current->data, idUser, username);
                current->data.status = 0; // Update status in memory
            }
            return;
        }
        current = current->next;
    }

    if (!found)
    {
        printf("Buku dengan judul \"%s\" tidak ditemukan.\n", target);
    }
}
```

- ii. searchInHeap() pada fungsi ini akan jalan ketika user memilih di Heap. Berbeda dengan Linear, di Heap searching akan menggunakan ID buku oleh karena itu fungsi ini akan membandingkan semua ID sampai ketemu ID yang sama dengan inputan User.

```
● ● ●

// cari buku dalam heap berdasarkan ID
void searchInHeap(struct heapBuku* heap, int idBuku) {
    int found = 0;
    for (int i = 0; i < heap->size; i++) {
        if (heap->data[i].idBuku == idBuku) {
            printf("\n==== BUKU DITEMUKAN ====\n");
            tampilPerBuku(&heap->data[i]);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Buku dengan ID %d tidak ditemukan dalam heap.\n", idBuku);
    }

    printf("\nTekan Enter untuk melanjutkan...");
    getchar();
}
```

b. Binary Search

Binary Search adalah pencarian yang akan mencari dan membandingkan elemen tengah lalu akan mencari ke kiri atau ke kanan sampai menemukan elemen yang dicari. Pada program kami Binary Search akan mencari dengan menggunakan ID buku.

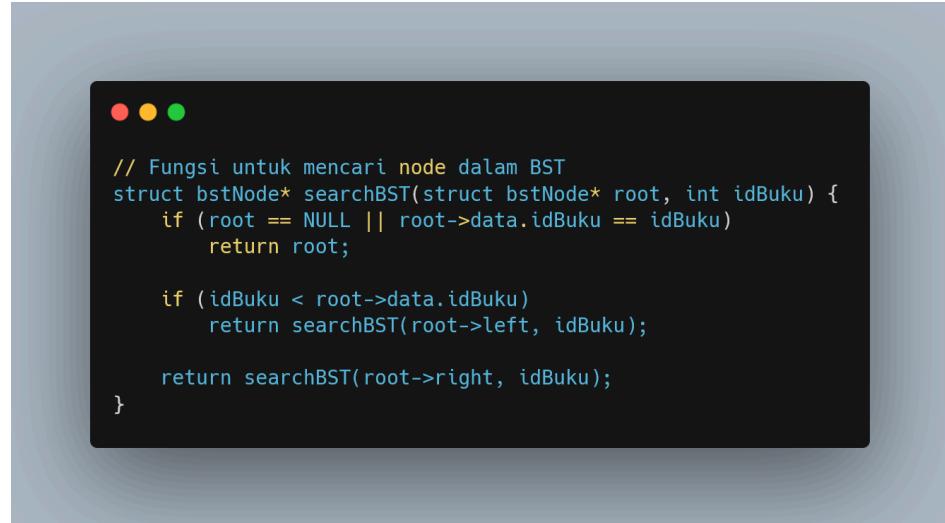
- i. cariBukuBST() fungsi ini berguna untuk User memasukan ID Buku lalu ID Buku akan dikirim ke searchBST().

```
● ● ●

void cariBukuBST(struct bstNode* root) {
    int idCari;
    printf("Masukkan ID Buku yang dicari: ");
    scanf("%d", &idCari);
    while (getchar() != '\n');

    struct bstNode* hasil = searchBST(root, idCari);
    if (hasil != NULL) {
        printf("\n==== BUKU DITEMUKAN ====\n");
        tampilPerBuku(&hasil->data);
    } else {
        printf("Buku dengan ID %d tidak ditemukan.\n", idCari);
    }
}
```

- ii. searchBST() fungsi ini berguna untuk menyari ID yang sudah di input oleh user dengan mengambil elemen tengah lalu akan dicari ke kiri atau ke kanan.



```
// Fungsi untuk mencari node dalam BST
struct bstNode* searchBST(struct bstNode* root, int idBuku) {
    if (root == NULL || root->data.idBuku == idBuku)
        return root;

    if (idBuku < root->data.idBuku)
        return searchBST(root->left, idBuku);

    return searchBST(root->right, idBuku);
}
```

c. Hasil Searching setiap Algoritma

1. Linear Search di Linear

```
linear.3
Masukkan judul buku yang dicari: rindu

==== Buku Ditemukan ====
Judul      : Rindu
Pengarang: Tere Liye
Tahun      : 2011
Status     : Tersedia
=====
Pilih aksi: (o) Ambil Buku | (q) Kembali Ke Home
Pilihan Anda: █
```

2. Linear Search di Heap

```
==== MENU HEAP ====
1. Tampilkan Heap
2. Cari Buku
3. Tambah Buku
4. Extract Buku
0. Kembali ke Home
Pilihan: 2
Masukkan ID buku yang dicari: 14

==== BUKU DITEMUKAN ===

==== Buku 14 ===
Judul      : Rectoverso
Pengarang: Dee Lestari
Tahun      : 2008
Status     : Tersedia
=====
Tekan Enter untuk melanjutkan...
```

3. Binary Search di BST

```
==== MENU BST ====
1. Lihat Semua Buku (Inorder)
2. Cari Buku
3. Tambah Buku
4. Hapus Buku
0. Home
Pilihan: 2
Masukkan ID Buku yang dicari: 13

==== BUKU DITEMUKAN ===

==== Buku 13 ===
Judul      : Partikel
Pengarang: Dee Lestari
Tahun      : 2012
Status     : Tersedia
=====
```

Evaluasi Kelompok

1. Felix Fernando Williams Lim : Menggambar struktur heap yang digunakan program. Evaluasi dari saya terhadap teman sekelompok sangat baik, Gading berinisiatif pertama kali untuk memulai penggerjaan UAS yang kami gunakan sebagai prototype pertama program yang telah dibuat dan juga membuat skema visualisasi tree dan alur program untuk penggunaan BST. Rafael kemudian menambahkan fitur-fitur yang menggunakan implementasi Heap. Naufal juga membantu dengan membuat isi laporan.
2. Naufal Rabbani : Membuat laporan dari Binary Search Tree, Binary Heap, Sorting algorithm, dan Searching algorithm. Evaluasi saya untuk teman kelompok sangatlah baik dan keren, semua berinisiatif untuk mengerjakan tugas seperti Putra langsung mengerjakan programnya lalu membagi tugas kita, Rafael yang mengerjakan tugasnya dengan baik dan tepat waktu, dan juga Felix yang sudah mengambar semua alur algoritmanya satu per satu dengan rapih dan mudah dipahami.
3. Rafael Romelo Gibran : Saya membuat menu untuk nomor 2 yaitu membuat heap pada program perpustakaan ini, evaluasi saya kepada teman-teman saya tidak ada karena menurut saya teman-teman saya yaitu felix, naufal, dan juga gading telah berhasil menyelesaikan tugasnya masing-masing dengan baik.
4. Gading Kelana Putra : Untuk UAS ini, saya mengerjakan codingan BST pada program perpustakaan khususnya bagian insert, search, dan delete (baik node dengan 1 anak maupun 2 anak). Saya juga membuat diagram BST untuk memvisualisasikan setiap eksekusi program. Pembagian tugas dengan Felix, Naufal, dan Rafael saya lakukan secara adil, dimana mereka mengerjakan nomor-nomor selanjutnya sekaligus mengevaluasi pekerjaan saya. Saya sangat nyaman bekerja dengan tim ini karena mereka semua inisiatif dan kooperatif.