**Nama : Andi Suci Khairunnisa**

**NIM : D121241085**

## 1. Latihan

Misalkan Anda diberikan sebuah buku telepon yang berisikan beberapa kerabat anda. Anda sedang ingin menelpon beberapa orang dan anda sedang terburu-buru untuk segera menelponnya. Ternyata, Anda tidak perlu mencari nama satu per satu, halaman demi halaman. Anda dapat mencari nama tersebut cukup dalam beberapa perbandingan. Buku telepon memiliki sifat khusus, yaitu terurut berdasarkan abjad. Dengan sifat ini, Anda bisa membuka halaman tengah dari buku telepon, lalu periksa apakah nama yang Anda cari ada pada halaman tersebut. Jika nama yang Anda cari berada di sebelah kiri halaman tengah, Anda akan melanjutkan pencarian hanya di bagian kiri, dan sebaliknya jika berada di sebelah kanan. Dengan cara ini, setiap perbandingan akan mengeliminasi separuh rentang pencarian.

Buatlah program dalam C untuk mengimplementasikan algoritma Binary Search yang dapat mencari nama dalam daftar nama orang pada buku telepon yang terurut berdasarkan abjad. Program harus menampilkan apakah nama tersebut ditemukan atau tidak, beserta indeksnya jika ditemukan.

**Daftar Nama:**

string names[] = { "Ahmad", "Alif", "Bella", "Budi", "Chandra", "Citra", "Diana", "Dewi", "Eli", "Eko", "Fajar", "Farhan", "Gani", "Gita", "Hani", "Hendra", "Indra", "Ika", "Jasmine", "Joko", "Kirana", "Kevin", "Lara", "Laras", "Lina", "Marcel", "Maya", "Nadia", "Nina", "Oki", "Omar", "Putri", "Qiana", "Rina", "Rudi", "Sari", "Sinta", "Tari", "Tina", "Uli", "Umar", "Vina", "Vira", "Wanda", "Wira", "Xena", "Yani", "Yudi", "Zaki"};

**Format Masukan dan Keluaran:**

Baris pertama merupakan n test case Baris kedua hingga ke baris-n merupakan nama yang dicari Keluaran terdiri dari n-baris yang merupakan indeks dari nama yang dicari (keluarkan “tidak ditemukan” jika nama yang dicari tidak terdapat pada data buku telepon yang telah diberikan)

**Jawaban:**

```c
#include <stdio.h>
#include <string.h>

// Fungsi untuk melakukan binary search
int binarySearch(char names[][20], int size, char target[]) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        // Jika nama ditemukan
        if (strcmp(names[mid], target) == 0) {
            return mid; // Mengembalikan indeks
        }
        // Jika target lebih kecil dari nama di tengah, cari di sebelah kiri
        else if (strcmp(names[mid], target) > 0) {
            right = mid - 1;
        }
        // Jika target lebih besar dari nama di tengah, cari di sebelah kanan
        else {
            left = mid + 1;
        }
    }
    return -1; // Mengembalikan -1 jika tidak ditemukan
}

int main() {
    char names[][20] = { "Ahmad", "Alif", "Bella", "Budi", "Chandra", "Citra", "Diana", "Dewi", "Eli", "Eko",
                         "Fajar", "Farhan", "Gani", "Gita", "Hani", "Hendra", "Indra", "Ika", "Jasmine",
                         "Joko", "Kirana", "Kevin", "Lara", "Laras", "Lina", "Marcel", "Maya", "Nadia",
                         "Nina", "Oki", "Omar", "Putri", "Qiana", "Rina", "Rudi", "Sari", "Sinta", "Tari",
                         "Tina", "Uli", "Umar", "Vina", "Vira", "Wanda", "Wira", "Xena", "Yani", "Yudi",
                         "Zaki" };
    int size = sizeof(names) / sizeof(names[0]);

    int n;
    printf("Masukkan jumlah test case: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        char target[20];
        printf("Masukkan nama yang dicari: ");
        scanf("%s", target);

        int result = binarySearch(names, size, target);
        if (result != -1) {
            printf("Nama ditemukan pada indeks: %d\n", result);
        } else {
            printf("Tidak ditemukan\n");
        }
    }

    return 0;
}
```

## 2. Tantangan

a. Binary Search

https://tlx.toki.id/courses/competitive 1/chapters/03/problems/E

**Deskripsi**

Pak Dengklek memiliki N ekor bebek. Bebek ke-i memiliki berat Ai. Pak Dengklek juga memiliki Q buah pertanyaan. Pertanyaan ke-ii berbunyi: berapa banyak bebek yang memiliki berat lebih dari (>) xi dan kurang dari sama dengan (≤) yi? Jawablah pertanyaan-pertanyaan tersebut!

**Masukan**

Masukan diberikan dalam format berikut:

N A1 A2 … AN

Q

x1 y1

x2 y2

⋮

xQ yQ

**Format Keluaran**

Untuk setiap pertanyaan, keluarkan sebuah baris berisi banyaknya bebek yang dimaksud.

**Jawaban:**

```c
#include <stdio.h>
#include <stdlib.h>

// Fungsi untuk membandingkan dua elemen (digunakan dalam qsort)
int banding(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

// Fungsi untuk mencari indeks pertama yang lebih dari nilai target
int cariLebihDari(int *bebek, int n, int target) {
    int kiri = 0, kanan = n - 1;
    while (kiri <= kanan) {
        int tengah = kiri + (kanan - kiri) / 2;
        if (bebek[tengah] > target) {
            kanan = tengah - 1; // Cari di sebelah kiri
        } else {
            kiri = tengah + 1; // Cari di sebelah kanan
        }
    }
    return kiri; // Kembali ke indeks pertama yang lebih dari target
}

// Fungsi untuk mencari indeks terakhir yang kurang dari sama dengan nilai target
int cariKurangSamaDengan(int *bebek, int n, int target) {
    int kiri = 0, kanan = n - 1;
    while (kiri <= kanan) {
        int tengah = kiri + (kanan - kiri) / 2;
        if (bebek[tengah] <= target) {
            kiri = tengah + 1; // Cari di sebelah kanan
        } else {
            kanan = tengah - 1; // Cari di sebelah kiri
        }
    }
    return kanan; // Kembali ke indeks terakhir yang kurang dari sama dengan target
}

int main() {
    int n, q;

    // Membaca jumlah bebek
    printf("Masukkan jumlah bebek: "); scanf("%d", &n);
    int bebek[n]; // Menggunakan array statis

    // Membaca berat bebek
    for (int i = 0; i < n; i++) {
        printf("berat bebek ke %d: ", i+1) ; scanf("%d", &bebek[i]);
    }

    // Mengurutkan berat bebek
    qsort(bebek, n, sizeof(int), banding);

    // Membaca jumlah pertanyaan
    printf ("Masukkan jumlah pertanyaan:"); scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        int x, y;
        printf("Masukkan batas yang diinginkan [batas_1 (spasi) batas_2: ") ; scanf("%d %d", &x, &y);

        // Mencari indeks untuk pertanyaan
        int indeksLebihDari = cariLebihDari(bebek, n, x);
        int indeksKurangSamaDengan = cariKurangSamaDengan(bebek, n, y);

        // Menghitung jumlah bebek yang memenuhi syarat
        int jumlahBebek = indeksKurangSamaDengan - indeksLebihDari + 1;

        // Menampilkan hasil
        printf("%d\n", jumlahBebek);
    }

    return 0;
}
```

**b. Sequantial Search**

**Deskripsi**

Pak Dengklek memiliki data yang terdiri atas N buah bilangan bulat: A1 hingga AN. Ia juga memiliki sebuah bilangan bulat X. Ia ingin tahu, di antara data tersebut, bilangan mana yang selisihnya dengan X paling kecil?

**Masukan**

Masukan diberikan dalam format berikut:

N X

A1 A2 … AN

**Keluaran**

Keluarkan sebuah baris berisi sebuah bilangan bulat dari data Pak Dengklek yang memiliki selisih terkecil dengan X.

**Jawaban**:

```c
#include <stdio.h>
#include <stdlib.h> // Untuk fungsi abs()

int main() {
    int n, x;

    // Membaca jumlah bilangan dan nilai X
    printf("Masukkan jumlah bilangan bulat dan nilai x [jumlah_bilangan_bulat (spasi) nilai_x]: \n") ;
    scanf("%d %d", &n, &x);
    int data[n]; // Menggunakan array statis untuk menyimpan data

    // Membaca bilangan bulat ke dalam array
    for (int i = 0; i < n; i++) {
       printf ("Masukkan bilangan bulat ke-%d: ", i+1); scanf("%d", &data[i]);
    }

    // Variabel untuk menyimpan bilangan dengan selisih terkecil
    int bilanganTerkecil = data[0];
    int selisihTerkecil = abs(data[0] - x); // Menghitung selisih awal

    // Melakukan pencarian secara sequential
    for (int i = 1; i < n; i++) {
        int selisih = abs(data[i] - x); // Menghitung selisih dengan X
        if (selisih < selisihTerkecil) {
            selisihTerkecil = selisih; // Update selisih terkecil
            bilanganTerkecil = data[i]; // Update bilangan terkecil
        }
    }

    // Menampilkan hasil
    printf("%d\n", bilanganTerkecil);
    return 0;
}
```