LAPORAN PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

TUGAS OTH WEEK 4 : ARRAY, POINTER, DAN FUNGSI

Di susun oleh :

CAHYANING ERDINIRA WIDIYA LESTARI

1203220089

IF 03-02

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY SURABAYA

2024

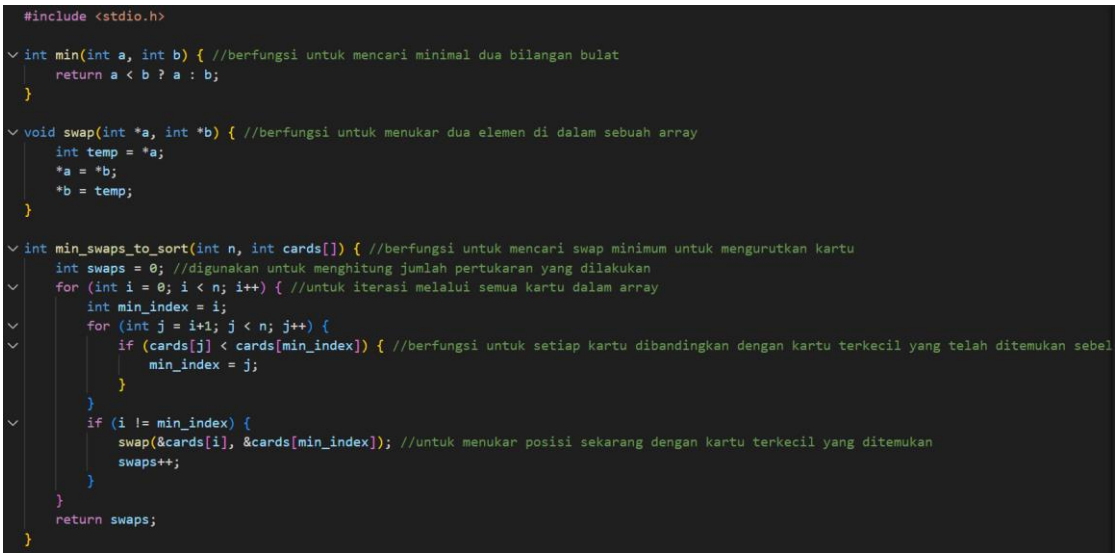
TABEL FORM PENILAIAN

Komponen Penilaian	Ya	Tidak
Soal 1 sesuai dengan output yang diinginkan		
Soal 2 sesuai dengan output yang diinginkan		
Bonus soal 1 dikerjakan		

SOAL 1

1. SOURCE CODE

- **Screenshoot Source Code**



```c
#include <stdio.h>

int min(int a, int b) { //berfungsi untuk mencari minimal dua bilangan bulat
    return a < b ? a : b;
}

void swap(int *a, int *b) { //berfungsi untuk menukar dua elemen di dalam sebuah array
    int temp = *a;
    *a = *b;
    *b = temp;
}

int min_swaps_to_sort(int n, int cards[]) { //berfungsi untuk mencari swap minimum untuk mengurutkan kartu
    int swaps = 0; //digunakan untuk menghitung jumlah pertukaran yang dilakukan
    for (int i = 0; i < n; i++) { //untuk iterasi melalui semua kartu dalam array
        int min_index = i;
        for (int j = i+1; j < n; j++) {
            if (cards[j] < cards[min_index]) { //berfungsi untuk setiap kartu dibandingkan dengan kartu terkecil yang telah ditemukan sebel
                min_index = j;
            }
        }
        if (i != min_index) {
            swap(&cards[i], &cards[min_index]); //untuk menukar posisi sekarang dengan kartu terkecil yang ditemukan
            swaps++;
        }
    }
    return swaps;
}
```

```c
int main() {
    int n;
    scanf("%d", &n); //untuk menginput jumlah kartu
    int cards[n];
    for (int i = 0; i < n; i++) {
        char card[3];
        scanf("%s", card); //untuk menginput nilai kartu
        if (card[0] >= '1' && card[0] <= '9') {
            cards[i] = card[0] - '0';
        } else {
            switch(card[0]) { //mengonversi J, Q, K menjadi nilai numerik
                case 'J':
                    cards[i] = 11;
                    break;
                case 'Q':
                    cards[i] = 12;
                    break;
                case 'K':
                    cards[i] = 13;
                    break;
            }
        }
    }
    printf("%d\n", min_swaps_to_sort(n, cards)); //output jumlah minimal langkah pertukaran
    return 0;
}
```

- **Source Code Serta Penjelasan**

```
#include <stdio.h>

int min(int a, int b) { //berfungsi untuk mencari minimal dua bilangan bulat
    return a < b ? a : b;
}

void swap(int *a, int *b) { //berfungsi untuk menukar dua elemen di dalam sebuah array
    int temp = *a;
    *a = *b;
    *b = temp;
}

int min_swaps_to_sort(int n, int cards[]) { //berfungsi untuk mencari swap minimum untuk mengurutkan kartu
    int swaps = 0; //digunakan untuk menghitung jumlah pertukaran yang dilakukan
    for (int i = 0; i < n; i++) { //untuk iterasi melalui semua kartu dalam array
        int min_index = i;
        for (int j = i+1; j < n; j++) {
            if (cards[j] < cards[min_index]) { //berfungsi untuk setiap kartu dibandingkan dengan kartu terkecil yang telah ditemukan sebelumnya. Jika kartu yang sedang dipertimbangkan lebih kecil dari kartu terkecil yang
```

```
ditemukan sebelumnya, maka min_index diubah menjadi indeks kartu yang
lebih kecil tersebut
                min_index = j;
            }
        }
        if (i != min_index) {
            swap(&cards[i], &cards[min_index]); //untuk menukar posisi sekarang
dengan kartu terkecil yang ditemukan
            swaps++;
        }
    }
    return swaps;
}

int main() {
    int n;
    scanf("%d", &n); //untuk menginput jumlah kartu
    int cards[n];
    for (int i = 0; i < n; i++) {
        char card[3];
        scanf("%s", card); //untuk menginput nilai kartu
        if (card[0] >= '1' && card[0] <= '9') {
            cards[i] = card[0] - '0';
        } else {
            switch(card[0]) { //mengonversi J, Q, K menjadi nilai numerik
                case 'J':
                    cards[i] = 11;
                    break;
                case 'Q':
                    cards[i] = 12;
                    break;
                case 'K':
                    cards[i] = 13;
                    break;
            }
        }
    }
    printf("%d\n", min_swaps_to_sort(n, cards)); //output jumlah minimal
langkah pertukaran
    return 0;
}
```

- **Screenshoot Source Code Bonus Soal 1**

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

int getCardValue(char card) { //berfungsi untuk mendapatkan nilai angka dari kartu
    if (card == 'J') return 11; //jika karakter kartu 'J' maka fungsi mengembalikan nilai 11 karena dalam aturan perm
    else if (card == 'Q') return 12;
    else if (card == 'K') return 13;
    else if (card == '1') return 10;
    else return (int)(card - '0'); //untuk mengembalikan nilai numerik dari karakter
}

void printCards(char *cards, int length) { //berfungsi untuk menampilkan urutan kartu
    for (int a = 0; a < length; a++) { //untuk iterasi melalui setiap elemen dalam array 'cards' dan variabel loop 'a
        printf("%c ", cards[a]); //untuk mencetak karakter yang ada di indeks ke 'a' dari array 'cards' untuk mencetak
    }
    printf("\n");
}
```

```c
int sortCards(char *cards, int length) { //berfungsi untuk mengurutkan kartu
    int swaps = 0; //untuk menghitung jumlah pertukaran yang dilakukan untuk mengurutkan kartu
    for (int a = 0; a < length - 1; a++) { //loop ini akan berhenti sebelum elemen terakhir karena tidak perlu memeri
        int min_idx = a; //'a' adalah indeks kartu terkecil yang ditemukan dalam iterasi saat ini
        for (int b = a; b < length; b++) {
            if (getCardValue(cards[b]) < getCardValue(cards[min_idx])) { //mengkonversi kartu ke nilai angka untuk me
                min_idx = b;
            }
        }
        if (min_idx != a) //berfungsi untuk pengecekan apakah kartu terkecil ditemukan pada indeks yang berbeda pada
        {
            char temp = cards[a]; //digunakan untuk menyimpan nilai sementara dari kartu pada posisi saat ini
            cards[a] = cards[min_idx]; //nilai kartu pada posisi 'min_idx' dipindahkan ke posisi saat ini 'a'
            cards[min_idx] = temp; //Nilai kartu yang disimpan dalam 'temp' dipindahkan ke posisi yang sebelumnya dii
            swaps++; //jumlah pertukaran ditambah satu setiap kali pertukaran dilakukan

            printf("Pertukaran %d : ", swaps); //untuk mengeluarkan output pertukaran
            printCards(cards, length); //setelah pertukaran dilakukan, urutan kartu baru dicetak ke layar menggunakan
        }

    }
    return swaps; //mengembalikan jumlah total pertukaran yang dilakukan
}
```

```c
int main() {
    int n; //mendeklarasikan untuk menyimpan jumlah kartu yang dimasukkan
    scanf("%d", &n); //untuk menginputkan jumlah kartu dan disimpan dalam variabel 'n'

    char cards[n]; //mendeklarasikan untuk menyimpan nilai-nilai kartu dan ukuran array ini ditentukan oleh jumlah

    for (int a = 0; a < n; a++) { //digunakan untuk mengisi array 'cards' dengan nilai-nilai kartu yang dimasukkan
        scanf(" %c", &cards[a]); //setiap kartu yang dimasukkan akan disimpan di dalam array 'cards'
    }

    int swaps = sortCards(cards, n); //berfungsi untuk mengurutkan kartu dalam array

    printf("%d\n", swaps); //mengoutputkan jumlah pertukaran kartu

    free(cards);
    return 0;
}
```

- **Source Code Bonus Soal 1 Serta Penjelasan**

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

int getCardValue(char card) { //berfungsi untuk mendapatkan nilai angka dari kartu
   if (card == 'J') return 11; //jika karakter kartu 'J' maka fungsi mengembalikan nilai 11 karena dalam aturan permainan 'j' mempunyai nilai 11
   else if (card == 'Q') return 12;
   else if (card == 'K') return 13;
   else if (card == '1') return 10;
   else return (int)(card - '0'); //untuk mengembalikan nilai numerik dari karakter
}

void printCards(char *cards, int length) { //berfungsi untuk menampilkan urutan kartu
   for (int a = 0; a < length; a++) { //untuk iterasi melalui setiap elemen dalam array 'cards' dan variabel loop 'a' diinisialisasi dengan nilai 0 dan ditingkatkan seiring dengan setiap iterasi hingga mencapai panjang array 'cards'
      printf("%c ", cards[a]); //untuk mencetak karakter yang ada di indeks ke 'a' dari array 'cards' untuk mencetak nilai kartu di posisi tersebut
   }
   printf("\n");
}

int sortCards(char *cards, int length) { //berfungsi untuk mengurutkan kartu
   int swaps = 0; //untuk menghitung jumlah pertukaran yang dilakukan untuk mengurutkan kartu
   for (int a = 0; a < length - 1; a++) { //loop ini akan berhenti sebelum elemen terakhir karena tidak perlu memeriksa elemen terakhir saat iterasi terakhir
      int min_idx = a; //'a' adalah indeks kartu terkecil yang ditemukan dalam iterasi saat ini
      for (int b = a; b < length; b++) {
         if (getCardValue(cards[b]) < getCardValue(cards[min_idx])) { //mengkonversi kartu ke nilai angka untuk membandingkan
            min_idx = b;
         }
      }
      if (min_idx != a) //berfungsi untuk pengecekan apakah kartu terkecil ditemukan pada indeks yang berbeda pada saat ini, jika sama ada kartu yang lebih kecil dari kartu saat ini, maka pertukaran dilakukan
      {
```

```
            char temp = cards[a]; //digunakan untuk menyimpan nilai sementara
dari kartu pada posisi saat ini
            cards[a] = cards[min_idx]; //nilai kartu pada posisi 'min_idx'
dipindahkan ke posisi saat ini 'a'
            cards[min_idx] = temp; //Nilai kartu yang disimpan dalam 'temp'
dipindahkan ke posisi yang sebelumnya diisi oleh nilai kartu 'cards[min_idx]',
di kode ini menyelesaikan proses pertukaran
            swaps++; //jumlah pertukaran ditambah satu setiap kali pertukaran
dilakukan

            printf("Pertukaran %d : ", swaps); //untuk mengeluarkan output
pertukaran
            printCards(cards, length); //setelah pertukaran dilakukan, urutan kartu
baru dicetak ke layar menggunakan fungsi 'printCards'
        }

    }
    return swaps; //mengembalikan jumlah total pertukaran yang dilakukan
}

int main() {
    int n; //mendeklarasikan untuk menyimpan jumlah kartu yang dimasukkan
    scanf("%d", &n); //untuk menginputkan jumlah kartu dan disimpan dalam
variabel 'n'

    char cards[n]; //mendeklarasikan untuk menyimpan nilai-nilai kartu dan
ukuran array ini ditentukan oleh jumlah kartu yang dimasukkan

    for (int a = 0; a < n; a++) { //digunakan untuk mengisi array 'cards' dengan
nilai-nilai kartu yang dimasukkan
        scanf(" %c", &cards[a]); //setiap kartu yang dimasukkan akan disimpan
di dalam array 'cards'
    }

    int swaps = sortCards(cards, n); //berfungsi untuk mengurutkan kartu dalam
array

    printf("%d\n", swaps); //mengoutputkan jumlah pertukaran kartu

    free(cards);
    return 0;
}
```

2. OUTPUT

- **Contoh Output 1**

```
4
6 6 9 7
1
```

- **Contoh Output 2**

```
5
3 2 8 7 4
2
```

- **Contoh Output 3**

```
6
10 J K Q 3 2
4
```

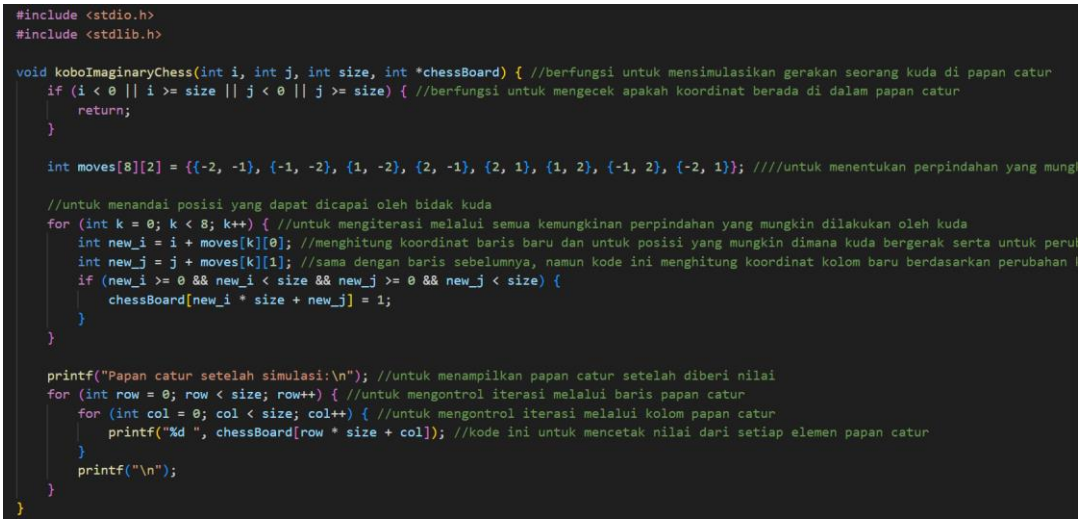
- **Contoh Output Bonus Soal 1**

```
8
9 4 2 J K 8 4 Q
Pertukaran 1 : 2 4 9 J K 8 4 Q
Pertukaran 2 : 2 4 4 J K 8 9 Q
Pertukaran 3 : 2 4 4 8 K J 9 Q
Pertukaran 4 : 2 4 4 8 9 J K Q
Pertukaran 5 : 2 4 4 8 9 J Q K
5
```

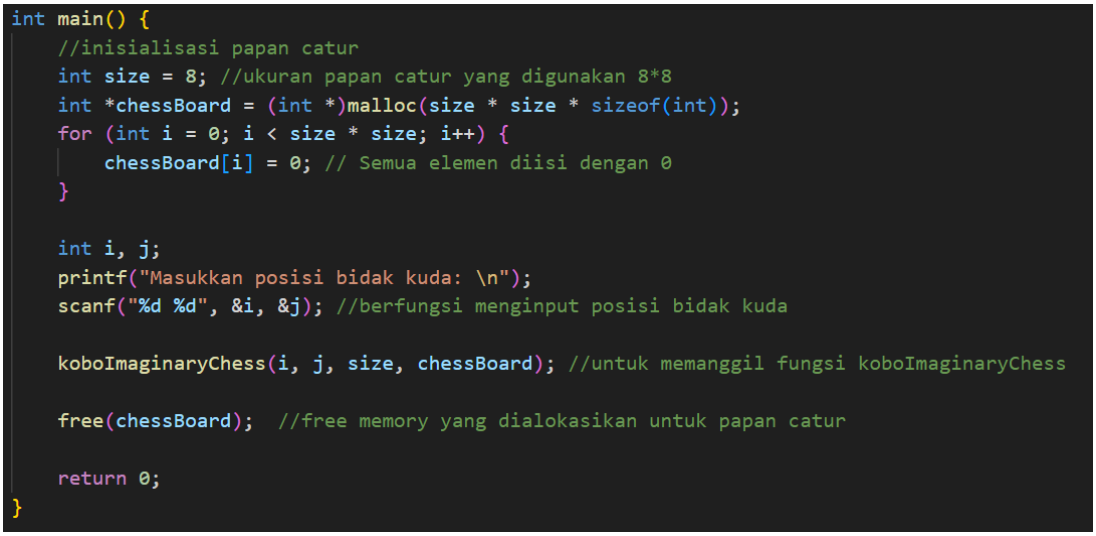
SOAL 2

1. SOURCE CODE

- **Screenshoot Source Code**



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

void koboImaginaryChess(int i, int j, int size, int *chessBoard) { //berfungsi untuk mensimulasikan gerakan seorang kuda di papan catur
    if (i < 0 || i >= size || j < 0 || j >= size) { //berfungsi untuk mengecek apakah koordinat berada di dalam papan catur
        return;
    }

    int moves[8][2] = {{-2, -1}, {-1, -2}, {1, -2}, {2, -1}, {2, 1}, {1, 2}, {-1, 2}, {-2, 1}}; ////untuk menentukan perpindahan yang mungk

    //untuk menandai posisi yang dapat dicapai oleh bidak kuda
    for (int k = 0; k < 8; k++) { //untuk mengiterasi melalui semua kemungkinan perpindahan yang mungkin dilakukan oleh kuda
        int new_i = i + moves[k][0]; //menghitung koordinat baris baru dan untuk posisi yang mungkin dimana kuda bergerak serta untuk perub
        int new_j = j + moves[k][1]; //sama dengan baris sebelumnya, namun kode ini menghitung koordinat kolom baru berdasarkan perubahan
        if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
            chessBoard[new_i * size + new_j] = 1;
        }
    }

    printf("Papan catur setelah simulasi:\n"); //untuk menampilkan papan catur setelah diberi nilai
    for (int row = 0; row < size; row++) { //untuk mengontrol iterasi melalui baris papan catur
        for (int col = 0; col < size; col++) { //untuk mengontrol iterasi melalui kolom papan catur
            printf("%d ", chessBoard[row * size + col]); //kode ini untuk mencetak nilai dari setiap elemen papan catur
        }
        printf("\n");
    }
}
```



```c
int main() {
    //inisialisasi papan catur
    int size = 8; //ukuran papan catur yang digunakan 8*8
    int *chessBoard = (int *)malloc(size * size * sizeof(int));
    for (int i = 0; i < size * size; i++) {
        chessBoard[i] = 0; // Semua elemen diisi dengan 0
    }

    int i, j;
    printf("Masukkan posisi bidak kuda: \n");
    scanf("%d %d", &i, &j); //berfungsi menginput posisi bidak kuda

    koboImaginaryChess(i, j, size, chessBoard); //untuk memanggil fungsi koboImaginaryChess

    free(chessBoard);  //free memory yang dialokasikan untuk papan catur

    return 0;
}
```

- **Source Code Serta Penjelasan**

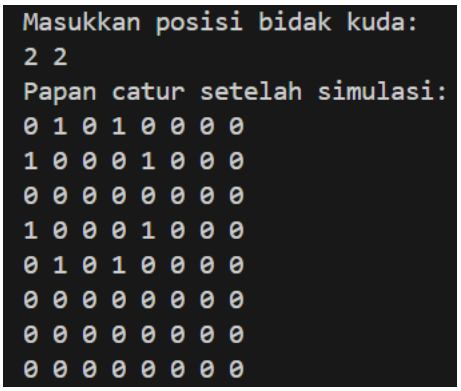
```
#include <stdio.h>
#include <stdlib.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) { //berfungsi
untuk mensimulasikan gerakan seorang kuda di papan catur
    if (i < 0 || i >= size || j < 0 || j >= size) { //berfungsi untuk mengecek apakah
koordinat berada di dalam papan catur
        return;
    }
```

```
    int moves[8][2] = {{-2, -1}, {-1, -2}, {1, -2}, {2, -1}, {2, 1}, {1, 2}, {-1, 2}, {-2, 1}}; ////untuk menentukan perpindahan yang mungkin dilakukan oleh bidak kuda

    //untuk menandai posisi yang dapat dicapai oleh bidak kuda
    for (int k = 0; k < 8; k++) { //untuk mengiterasi melalui semua kemungkinan perpindahan yang mungkin dilakukan oleh kuda
        int new_i = i + moves[k][0]; //menghitung koordinat baris baru dan untuk posisi yang mungkin dimana kuda bergerak serta untuk perubahan baris yang ditentukan
        int new_j = j + moves[k][1]; //sama dengan baris sebelumnya, namun kode ini menghitung koordinat kolom baru berdasarkan perubahan kolom yang ditentukan
        if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
            chessBoard[new_i * size + new_j] = 1;
        }
    }

    printf("Papan catur setelah simulasi:\n"); //untuk menampilkan papan catur setelah diberi nilai
    for (int row = 0; row < size; row++) { //untuk mengontrol iterasi melalui baris papan catur
        for (int col = 0; col < size; col++) { //untuk mengontrol iterasi melalui kolom papan catur
            printf("%d ", chessBoard[row * size + col]); //kode ini untuk mencetak nilai dari setiap elemen papan catur
        }
        printf("\n");
    }
}

int main() {
    //inisialisasi papan catur
    int size = 8; //ukuran papan catur yang digunakan 8*8
    int *chessBoard = (int *)malloc(size * size * sizeof(int));
    for (int i = 0; i < size * size; i++) {
        chessBoard[i] = 0; // Semua elemen diisi dengan 0
    }

    int i, j;
    printf("Masukkan posisi bidak kuda: \n");
    scanf("%d %d", &i, &j); //berfungsi menginput posisi bidak kuda

    koboImaginaryChess(i, j, size, chessBoard); //untuk memanggil fungsi koboImaginaryChess
```

```
    free(chessBoard);  //free memory yang dialokasikan untuk papan catur

    return 0;
}
```

2. OUTPUT

- **Hasil Output Posisi Bidak Kuda 2 2**



- **Hasil Output Posisi Bidak Kuda 3 7**