

NAMA LENGKAP: JOURDAN SOLITHIO POLLO MOLLE

KELAS :IF 03-02

NIM :1203230057

MATKUL :ALGORITMA DAN STRUKTUR DATA(Praktikum)

LAPORAN TUGAS PRAKTIKUM

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

1. SOURCE CODE

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

int get_card_value(char card) {
    if (card == 'J') return 11;
    else if (card == 'Q') return 12;
    else if (card == 'K') return 13;
    else if (card == '1') return 10;
    else return card - '0';
}

int main() {
    int n, i, j, min_swaps = 0;
    char cards[100];
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf(" %c", &cards[i]);
    }
    for (i = 0; i < n; i++) {
        int min_idx = i;
        for (j = i + 1; j < n; j++) {
```

```

        if (get_card_value(cards[j]) < get_card_value(cards[min_idx])) {
            min_idx = j;
        }
    }
    if (min_idx != i) {
        char temp = cards[i];
        cards[i] = cards[min_idx];
        cards[min_idx] = temp;
        min_swaps++;
    }
}
printf("%d\n", min_swaps);
return 0;
}

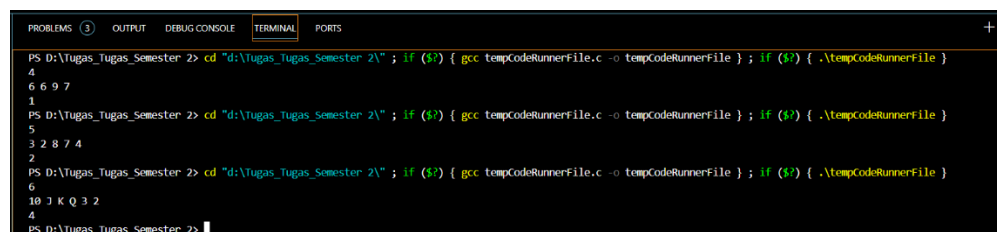
```

PENJELASAN

Program di atas merupakan sebuah program dalam bahasa C yang menghitung jumlah minimum pertukaran elemen yang diperlukan untuk mengurutkan array kartu dalam urutan non-menurun berdasarkan nilai kartu. Program ini menggunakan algoritma selection sort untuk mengurutkan array.

1. Fungsi `get_card_value` digunakan untuk mendapatkan nilai numerik dari kartu. Kartu 'J', 'Q', dan 'K' memiliki nilai masing-masing 11, 12, dan 13. Kartu '1' (sepuluh) memiliki nilai 10, sedangkan kartu numerik lainnya memiliki nilai yang sama dengan angka tersebut.
2. Program utama membaca input berupa jumlah kartu (`n`) dan array kartu (`cards`).
3. Setiap kartu dalam array dibaca dan diurutkan menggunakan algoritma selection sort. Algoritma ini mencari nilai minimum dari array yang belum diurutkan, lalu menukarnya dengan elemen pertama dari array yang belum diurutkan. Proses ini diulang hingga seluruh array terurut.
4. Selama proses pengurutan, program menghitung jumlah pertukaran minimum yang dilakukan (`min_swaps`).
5. Setelah selesai mengurutkan, program mencetak jumlah minimum pertukaran yang dilakukan (`min_swaps`).

OUTPUT PROGRAM



```

PROBLEMS (3) OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
4
6 6 9 7
1
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
5
3 2 8 7 4
2
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
6
10 J K Q 3 2
4
PS D:\Tugas_Tugas_Semester 2>

```

2. SOURCE CODE

```
#include <stdio.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Menginisialisasi semua elemen array dengan nilai 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Menghitung kemungkinan posisi yang dapat dilalui oleh bidak kuda
    int moves[8][2] = { {-2, -1}, {-2, 1}, {2, -1}, {2, 1}, {-1, -2}, {-1, 2},
{1, -2}, {1, 2} };
    for (int k = 0; k < 8; k++) {
        int newX = i + moves[k][0];
        int newY = j + moves[k][1];
        if (newX >= 0 && newX < size && newY >= 0 && newY < size) {
            *(chessBoard + newX * size + newY) = 1;
        }
    }

    // Menampilkan output array
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d ", *(chessBoard + x * size + y));
        }
        printf("\n");
    }
}

int main() {
    int i, j;
    scanf("%d %d", &i, &j);
    int chessBoard[8][8];
    koboImaginaryChess(i, j, 8, (int *)chessBoard);
    return 0;
}
```

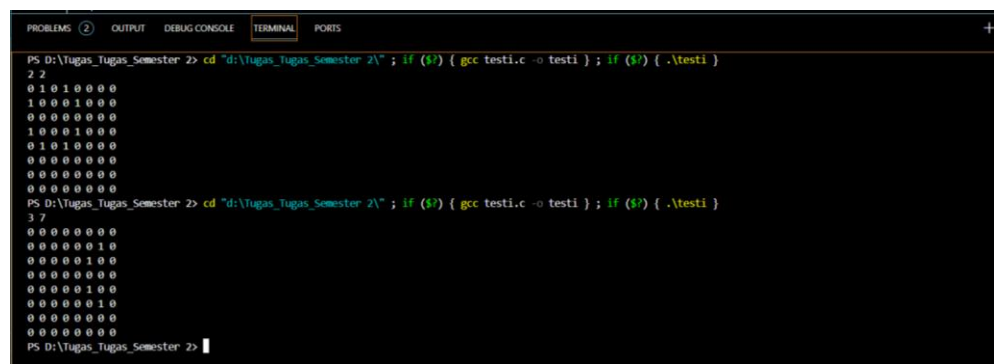
PENJELASAN

Program di atas adalah implementasi dalam bahasa C untuk mensimulasikan langkah-langkah yang dapat dilakukan oleh sebuah bidak kuda (knight) dalam catur. Pada awalnya, program akan meminta input posisi awal bidak kuda (dengan menganggap papan catur berukuran 8x8) dari pengguna. Kemudian, program akan menampilkan papan catur yang telah diisi dengan angka 1 pada posisi-posisi yang dapat dicapai oleh bidak kuda dalam sekali jalan, dan angka 0 pada posisi lainnya.

Langkah-langkah utama dalam program ini adalah sebagai berikut:

1. Fungsi `kobolMaginaryChess` menerima input posisi awal bidak kuda (`i` dan `j`), ukuran papan catur (`size`), dan array 2D (`chessBoard`) yang merepresentasikan papan catur. Pada awalnya, semua elemen array diinisialisasi dengan nilai 0.
2. Kemudian, program menghitung kemungkinan posisi yang dapat dilalui oleh bidak kuda berdasarkan posisi awalnya. Posisi-posisi yang mungkin dilalui ini ditandai dengan mengubah nilai elemen array yang bersesuaian menjadi 1.
3. Setelah itu, program menampilkan papan catur hasil simulasi dengan menampilkan nilai setiap elemen array ke layar.
4. Pada fungsi `main`, program meminta input posisi awal bidak kuda dari pengguna, kemudian membuat array 2D `chessBoard` berukuran 8x8 untuk digunakan dalam simulasi. Setelah itu, fungsi `kobolMaginaryChess` dipanggil dengan parameter yang sesuai untuk melakukan simulasi dan menampilkan papan catur hasilnya.

OUTPUT PROGRAM



```
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc testi.c -o testi } ; if ($?) { .\testi }
2 2
0 1 0 1 0 0 0 0
1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0
0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
PS D:\Tugas_Tugas_Semester 2> cd "d:\Tugas_Tugas_Semester 2\" ; if ($?) { gcc testi.c -o testi } ; if ($?) { .\testi }
3 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
PS D:\Tugas_Tugas_Semester 2> |
```