PENJELASAN CODINGAN ARRAY POINTER DAN FUNGSI

NAMA: WICIPTO SETIADI

NIM: 1203230042

KELAS: IF-03-01

KOMPONEN PENILAIAN	YA	TIDAK
Soal 1 seuai dengan output yang diinginkan	o	
Soal 2 sesuai dengan output yang diinginkan	o	
Bonus soal 1 kerjakan	o	

SOAL NOMER 1

(PEMROSESAN KARTU)

```
48      // Mengalokasikan memori untuk array kartu
49      int *cards = (int *)malloc(n * sizeof(int));
```

Array ini akan menyimpan nilai-nilai kartu yang dibaca dari input

```
55          if (strcmp(card, "J") == 0) {
56              cards[i] = 11;
57          } else if (strcmp(card, "Q") == 0) {
58              cards[i] = 12;
59          } else if (strcmp(card, "K") == 0) {
60              cards[i] = 13;
61          } else {
62              cards[i] = atoi(card);
63          }
64      }
```

Ketika membaca input, jika nilai kartu adalah ‘J’, ‘Q’, atau ‘K’, maka kita mengonversinya menjadi nilai yang sesuai, yaitu 11, 12, atau 13. Ini dilakukan agar semua nilai kartu dapat diurutkan dengan benar

(PERTUKARAN KARTU)

```
10      for (i = 0; i < n; i++) {
11          int min_index = i;
12          for (j = i+1; j < n; j++) {
13              if (cards[j] < cards[min_index]) {
14                  min_index = j;
15              }
16          }
17          if (min_index != i) {
18              // Melakukan pertukaran posisi kartu
19              int temp = cards[i];
20              cards[i] = cards[min_index];
21              cards[min_index] = temp;
22              swaps++;
```

Kita menggunakan algoritma pengurutan seleksi untuk mengurutkan kartu. Setiap kali menemukan kartu yang lebih kecil, kita menukar posisi kartu tersebut dengan kartu yang lebih kecil itu.

```
    for (int i = 0; i < n; i++) {
        if (cards[i] == 11) {
            printf("J ");
        } else if (cards[i] == 12) {
            printf("Q ");
        } else if (cards[i] == 13) {
            printf("K ");
        } else {
            printf("%d ", cards[i]);
        }
    }
```

Saat melakukan pertukaran, kita juga mencetak susunan kartu setelah pertukaran dilakukan.

```
    // Mencetak jumlah langkah minimum pertukaran
    printf("Jumlah langkah minimum pertukaran: %d\n", minimum_swaps);

```

Setelah semua pertukaran selesai, program akan mencetak jumlah angkah minimum pertukaran yang diperlukan.

```
    // Membebaskan memori yang dialokasikan untuk array kartu
    free(cards);
```

Terakhir, kita juga membebaskan memori yang telah dialokasikan untuk array kartu. Dengan angkah, program akan menerima input kartu, mengurutkannya, dan mencetak setiap angkah pertukaran serta jumlah angkah minimum pertukaran yang diperlukan.

OUTPUT

```
4
6 6 9 7
Susunan kartu sebelum pertukaran: 6 6 9 7
Pertukaran 1: 6 6 7 9
Jumlah langkah minimum pertukaran: 1
```

```
8
9 4 2 J K 8 4 Q
Susunan kartu sebelum pertukaran: 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
Jumlah langkah minimum pertukaran: 5
```

SOAL NOMER 2

```
void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
```

Deklarasi fungsi koboImaginaryChess yang menerima empat parameter:

- ‘I’ dan ‘j’: koordinat posisi awal bidak kuda.
- ‘size’: ukuran papan catur (dalam hal ini 8x8).
- *’chessBoard’: pointer ke array 2D yang merepresentasikan papan catur.

```
    for (int row = 0; row < size; row++) {
        for (int col = 0; col < size; col++) {
            *(chessBoard + row * size + col) = 0;
        }
    }
```

Inisialisasi papan catur dengan mengisi setiap elemen dengan nilai 0 menggunakan nested loop.

```
    int dx[] = {-2, -1, 1, 2, 2, 1, -1, -2};
    int dy[] = {1, 2, 2, 1, -1, -2, -2, -1};
```

Deklarasi array ‘dx’ dan ‘dy’ yang merepresentasikan perubahan koordinat horizontal dan vertikal yang mungkin dilakukan oleh bidak kuda.

```
    for (int k = 0; k < 8; k++) {
        int x = i + dx[k];
        int y = j + dy[k];

        if (x >= 0 && x < size && y >= 0 && y < size) {
            *(chessBoard + x * size + y) = 1;
        }
    }
```

Loop ini akan melakukan iterasi untuk semua kemungkinan gerakan bidak kuda dan menandai posisi-posisi yang dapat dicapai oleh bidak kuda dengan nilai 1 pada papan catur.

```
    for (int row = 0; row < size; row++) {
        for (int col = 0; col < size; col++) {
            printf("%d ", *(chessBoard + row * size + col));
        }
        printf("\n");
    }
}
```

Loop ini mencetak array yang merepresentasikan papan catur ke layar, sehingga kita dapat melihat posisi-posisi yang dapat dicapai oleh bidak kuda.

```
34  int main() {
35      int i, j;
36      scanf("%d %d", &i, &j);
```

Fungsi main sebagai entry point program. scanf digunakan untuk membaca input posisi awal bidak kuda dari pengguna.

```
38      int size = 8;
39      int *chessBoard = (int *)malloc(size * size * sizeof(int));
```

Deklarasi variabel size yang menunjukkan ukuran papan catur dan alokasi memori dinamis untuk array chessBoard dengan ukuran size x size.

```
41      if (chessBoard == NULL) {
42          printf("Memory allocation failed.");
43          return 1;
44      }
```

Pengecekan apakah alokasi memori berhasil. Jika gagal, program akan mencetak pesan kesalahan dan keluar dengan status error.

```
46      koboImaginaryChess(i, j, size, chessBoard);
```

Memanggil fungsi koboImaginaryChess untuk menandai posisi-posisi yang dapat dicapai oleh bidak kuda dan mencetak papan catur.

```
48      free(chessBoard);
49      return 0;
50  }
```

Menggunakan free untuk membebaskan memori yang dialokasikan untuk chessBoard dan mengembalikan nilai 0 untuk menandakan bahwa program berakhir dengan sukses.

OUTPUT

```
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
```

```
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
```