SATRIO DWI YANDA ARIFIN

1203230020

IF 03-01

TUGAS PRATIKUM

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

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void printCard(int number) {
    if (number == 1) printf("1 ");
    else if (number == 2) printf("2 ");
    else if (number == 3) printf("3 ");
    else if (number == 4) printf("4 ");
    else if (number == 5) printf("5 ");
    else if (number == 6) printf("6 ");
    else if (number == 7) printf("7 ");
    else if (number == 8) printf("8 ");
    else if (number == 9) printf("9 ");
    else if (number == 10) printf("10 ");
    else if (number == 11) printf("J ");
    else if (number == 12) printf("Q ");
    else if (number == 13) printf("K ");
    else printf("%d ", number);
}

void sortCards(int n, int *arr) {
    int swaps = 0;

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
                swaps++;

                printf("Langkah %d: Pertukaran antara kartu %d (%c) dan kartu %d (%c)\n",
swaps, j+1, arr[j], j+2, arr[j+1]);

                printf("Kartu setelah pertukaran %d: ", swaps);
```

```
            for (int k = 0; k < n; k++) {
                printCard(arr[k]);
            }
            printf("\n");
        }
    }
}

    printf("Jumlah pertukaran yang dilakukan : %d\n", swaps);
}

int main() {
    int n;
    printf("Masukkan jumlah kartu : ");
    scanf("%d", &n);

    int arr[n];
    char temp[3];
    printf("Masukkan angka kartu (1-10, J/Q/K) : ");
    for (int i = 0; i < n; i++) {
        scanf("%s", temp);
        if (strcmp(temp, "J") == 0) {
            arr[i] = 11;
        } else if (strcmp(temp, "Q") == 0) {
            arr[i] = 12;
        } else if (strcmp(temp, "K") == 0) {
            arr[i] = 13;
        } else {
            sscanf(temp, "%d", &arr[i]);
        }
    }

    sortCards(n, arr);

    return 0;
}
```

Hasil OUTPUT

```
Kartu setelah pertukaran 2: 10 J 9 K Q
Langkah 3: Pertukaran antara kartu 4 (
) dan kartu 5 (
Kartu setelah pertukaran 3: 10 J 9 Q K
Langkah 4: Pertukaran antara kartu 2 (  ) dan kartu 3 (
)
Kartu setelah pertukaran 4: 10 9 J Q K
Langkah 5: Pertukaran antara kartu 1 (  ) dan kartu 2 (
)
Kartu setelah pertukaran 5: 9 10 J Q K
Jumlah pertukaran yang dilakukan : 5
PS C:\Users\Satrio Dwi Y\Documents\kuliah>
```

PENJELASAN:

1.

```
void swap(int* a, int* b) {
   int temp = *a;
   *a = *b;
   *b = temp;
```

Fungsi SWAP adalah sebuah fungsi untuk menukar nilai dari dua variabel yang diberikan sebagai argumen. Fungsi ini menggunakan pointer untuk mengakses nilai variabel yang sebenarnya dan melakukan pertukaran nilai di antara mereka. Berikut adalah penjelasan singkat tentang cara kerjanya:

Fungsi SWAP menerima dua pointer ke integer (int* a dan int* b) sebagai argumen.
Variabel TEMP digunakan untuk menyimpan nilai dari variabel yang pertama kali ditunjuk oleh pointer A
Nilai dari variabel yang ditunjuk oleh A digantikan dengan nilai dari variabel yang ditunjuk oleh B
Nilai dari variabel yang ditunjuk oleh B digantikan dengan nilai yang disimpan di variabel TEMP

Dengan cara ini, nilai kedua variabel dapat saling dipertukarkan tanpa kehilangan data atau nilai aslinya. Fungsi ini sangat berguna dalam berbagai kasus, termasuk dalam pengurutan atau algoritma lain yang melibatkan pertukaran nilai antar variabel.

2.

```c
void printCard(int number) {
    if (number == 1) printf("1 ");
    else if (number == 2) printf("2 ");
    else if (number == 3) printf("3 ");
    else if (number == 4) printf("4 ");
    else if (number == 5) printf("5 ");
    else if (number == 6) printf("6 ");
    else if (number == 7) printf("7 ");
    else if (number == 8) printf("8 ");
    else if (number == 9) printf("9 ");
    else if (number == 10) printf("10 ");
    else if (number == 11) printf("J ");
    else if (number == 12) printf("Q ");
    else if (number == 13) printf("K ");
    else printf("%d ", number);
}
```

Fungsi printCard yang Anda tulis digunakan untuk mencetak nilai kartu berdasarkan angka yang diberikan sebagai argumen. Fungsi ini memeriksa nilai angka yang diberikan dan mencetak karakter yang sesuai dengan nilai tersebut.

Argumen Fungsi: Fungsi printCard menerima satu parameter bertipe integer yang mewakili nilai kartu yang akan dicetak.

```c
void printCard(int number
```

Pemilihan Karakter: Fungsi ini menggunakan struktur percabangan if-else untuk memeriksa nilai dari parameter number dan mencetak karakter yang sesuai dengan nilai tersebut.

c

```c
if (number == 1) printf("1 ");
    else if (number == 2) printf("2 ");
    else if (number == 3) printf("3 ");
    else if (number == 4) printf("4 ");
    else if (number == 5) printf("5 ");
    else if (number == 6) printf("6 ");
    else if (number == 7) printf("7 ");
    else if (number == 8) printf("8 ");
    else if (number == 9) printf("9 ");
    else if (number == 10) printf("10 ");
```

```
    else if (number == 11) printf("J ");
    else if (number == 12) printf("Q ");
    else if (number == 13) printf("K ");
    else printf("%d ", number)
```

Jika nilai number adalah 1 hingga 10, fungsi ini akan mencetak karakter angka yang sesuai. Jika nilai number adalah 11, 12, atau 13, fungsi akan mencetak karakter 'J', 'Q', atau 'K' secara berturut-turut. Jika nilai number tidak cocok dengan kasus-kasus di atas, maka fungsi akan mencetak nilai number itu sendiri.

3.

```
void sortCards(int n, int *arr) {
    int swaps = 0;
```

Fungsi **sortCards** adalah implementasi dari algoritma pengurutan bubble sort yang bertujuan untuk mengurutkan array **arr** yang berisi nilai-nilai kartu.

```
for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
                swaps++;
```

Loop bersarang dalam fungsi **sortCards** bertanggung jawab untuk mengurutkan elemen-elemen dalam array **arr** menggunakan algoritma bubble sort. Mari kita jelaskan langkah-langkahnya dengan lebih rinci:

Iterasi Pertama (Loop i)
Loop ini bertanggung jawab untuk mengatur iterasi pada setiap fase pengurutan. Setiap iterasi mengurangi jumlah elemen yang diakses, karena setiap iterasi membawa elemen terbesar ke posisi terakhir.
Karena kita ingin memastikan bahwa elemen terakhir setelah iterasi pertama adalah elemen terbesar, iterasi hanya berjalan sampai ke **n - 1**. Jika iterasi melebihi **n - 1**, akan ada akses ke elemen di luar array, yang tidak diinginkan.
Iterasi Kedua (Loop j)
Loop ini bertanggung jawab untuk membandingkan dan menukar elemen-elemen dalam array.
Karena pada setiap iterasi **i**, elemen terbesar akan berada di posisi terakhir, kita tidak perlu memeriksa elemen terakhir di setiap iterasi selanjutnya. Oleh karena itu, iterasi hanya berjalan sampai **n - i - 1**.
Misalnya, pada iterasi pertama, elemen terbesar telah berada di posisi terakhir, sehingga tidak perlu membandingkan elemen tersebut lagi pada iterasi selanjutnya. Oleh karena itu, iterasi kedua hanya akan berjalan hingga elemen kedua dari belakang.
Pertukaran Elemen

Setiap kali dua elemen berturut-turut tidak berada dalam urutan yang benar (elemen ke-i lebih besar dari elemen ke-(i+1)), mereka ditukar.
Fungsi **swap** dipanggil untuk menukar nilai kedua elemen yang tidak terurut tersebut.
Setiap kali pertukaran dilakukan, variabel **swaps** diinkremen untuk menghitung jumlah total pertukaran yang dilakukan selama proses pengurutan.
Visualisasi Langkah Pertukaran
Setelah setiap pertukaran, program mencetak pesan yang mencatat langkah pertukaran serta nilai-nilai kartu setelah pertukaran.
ini membantu pengguna memahami proses pengurutan dan memvisualisasikan langkah-langkah yang dilakukan oleh algoritma.

Dengan demikian, loop bersarang dalam fungsi **sortCards** bertanggung jawab untuk menjalankan algoritma bubble sort, yang secara bertahap memindahkan nilai-nilai kartu ke posisi yang tepat dalam array.

4.

```
printf("Langkah %d: Pertukaran antara kartu %d (%c) dan kartu %d (%c)\n", swaps, j+1, arr[j], j+2, arr[j+1]);

            printf("Kartu setelah pertukaran %d: ", swaps);
            for (int k = 0; k < n; k++) {
                printCard(arr[k]);
            }
            printf("\n");
```

Pesan Pertukaran Kartu:
Pesan ini mencetak langkah pertukaran antara dua kartu yang sedang diproses.
Menggunakan format **"Langkah %d: Pertukaran antara kartu %d (%c) dan kartu %d (%c)\n"**.
%d mencetak jumlah total pertukaran yang telah dilakukan.
%c mencetak karakter yang mewakili nilai kartu dalam bentuk teks.
j+1 dan **j+2** menunjukkan posisi dua kartu yang dipertukarkan.
arr[j] dan **arr[j+1]** adalah nilai-nilai kartu yang dipertukarkan.
Pesan Kartu Setelah Pertukaran:
Pesan ini mencetak nilai-nilai kartu dalam array setelah pertukaran tersebut.
Menggunakan format **"Kartu setelah pertukaran %d: "**.
%d mencetak jumlah total pertukaran yang telah dilakukan.
Melakukan iterasi melalui array untuk mencetak nilai setiap kartu setelah pertukaran.
Menggunakan fungsi **printCard** untuk mencetak nilai setiap kartu dalam bentuk teks.
Setelah semua nilai kartu dicetak, **printf("\n")** digunakan untuk mencetak baris baru.

Dengan pesan-pesan ini, pengguna dapat memahami dan melacak langkah-langkah pertukaran kartu yang terjadi selama proses pengurutan. Ini memberikan pemahaman yang lebih baik tentang bagaimana algoritma bubble sort berfungsi dalam mengurutkan array.

5.

```
int main() {
    int n;
    printf("Masukkan jumlah kartu : ");
    scanf("%d", &n);
```

Bagian ini adalah awal dari fungsi **main()**, yang bertanggung jawab untuk menjalankan program utama

Input Jumlah Kartu:
Pesan **"Masukkan jumlah kartu : "** dicetak ke layar menggunakan fungsi **printf()**.
Pengguna diminta untuk memasukkan jumlah kartu.
Nilai yang dimasukkan oleh pengguna disimpan dalam variabel **n** menggunakan fungsi **scanf()**.

6.

```
int arr[n];
    char temp[3];
    printf("Masukkan angka kartu (1-10, J/Q/K) : ");
    for (int i = 0; i < n; i++) {
        scanf("%s", temp);
        if (strcmp(temp, "J") == 0) {
            arr[i] = 11;
        } else if (strcmp(temp, "Q") == 0) {
            arr[i] = 12;
        } else if (strcmp(temp, "K") == 0) {
            arr[i] = 13;
        } else {
            sscanf(temp, "%d", &arr[i]);
        }
    }

    sortCards(n, arr);

    return 0;
}
```

Deklarasi Array arr:
Array **arr** dideklarasikan dengan ukuran **n**, yang merupakan jumlah kartu yang telah dimasukkan oleh pengguna.
Ini adalah array yang akan menyimpan nilai-nilai kartu yang dimasukkan.
Meminta Input Nilai Kartu:
Pesan **"Masukkan angka kartu (1-10, J/Q/K) : "** dicetak ke layar menggunakan **printf()** untuk meminta pengguna memasukkan nilai-nilai kartu.
Sebuah loop **for** digunakan untuk mengiterasi sebanyak **n** kali, sesuai dengan jumlah kartu yang diminta.

Membaca dan Menyimpan Nilai Kartu:
Setiap kali loop berjalan, nilai kartu dimasukkan oleh pengguna menggunakan **scanf()** dan disimpan dalam variabel **temp**.
Nilai-nilai kartu kemudian disimpan dalam array **arr**.
Jika nilai kartu adalah "J", "Q", atau "K", maka nilai tersebut diubah menjadi 11, 12, atau 13 sesuai dengan aturan yang telah ditentukan. Ini dilakukan dengan menggunakan fungsi **strcmp()** untuk memeriksa apakah nilai yang dimasukkan adalah "J", "Q", atau "K".
Jika nilai kartu bukan "J", "Q", atau "K", maka nilai tersebut dianggap sebagai angka, dan disalin ke dalam array tanpa perubahan.
Memanggil Fungsi sortCards:
Setelah semua nilai kartu dimasukkan dan disimpan dalam array, fungsi **sortCards** dipanggil dengan argumen **n** (jumlah kartu) dan array **arr**.
Fungsi ini akan mengurutkan nilai-nilai kartu yang disimpan dalam array **arr**.

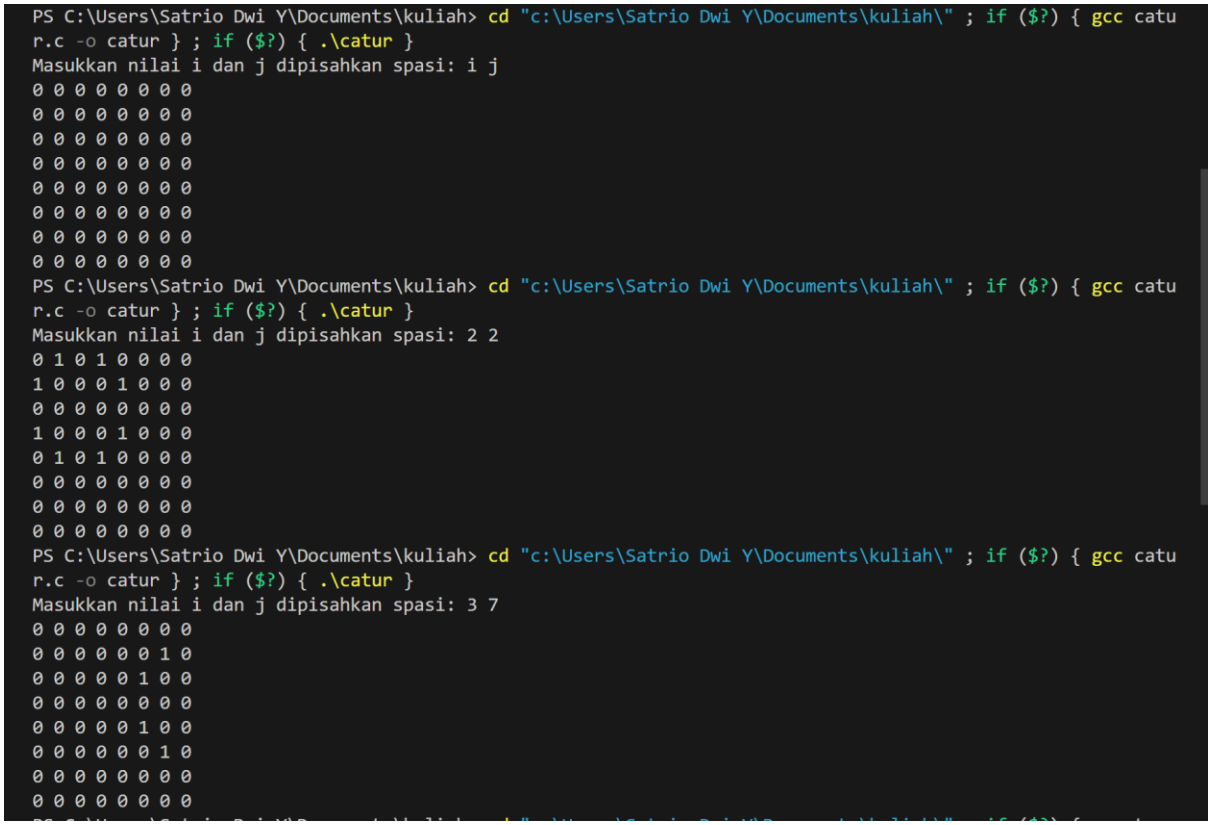
Mengembalikan Nilai 0:
Fungsi **main()** mengembalikan nilai 0 untuk menandakan bahwa program telah berjalan dengan sukses.

Dengan demikian, pada tahap ini, program meminta pengguna untuk memasukkan nilai-nilai kartu, menyimpannya dalam array, mengurutkannya menggunakan fungsi **sortCards**, dan kemudian mengakhiri eksekusi program.

SOAL 2

```c
#include <stdio.h>

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

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

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

    for (int k = 0; k < 8; k++) {
        int new_i = i + dx[k];
        int new_j = j + dy[k];

        if (new_i >= 0 && new_i < size && new_j >= 0 && new_j < size) {
            *(chessBoard + new_i*size + new_j) = 1;
        }
    }

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

int main() {
    int i, j;
    printf("Masukkan nilai i dan j dipisahkan spasi: ");
    scanf("%d %d", &i, &j);

    int chessBoard[8][8];
    koboImaginaryChess(i, j, 8, (int *)chessBoard);

    return 0;
}
```

HASIL OUTPUT



PENJELASAN:

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

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

Pada bagian ini, terdapat dua loop bersarang yang bertujuan untuk menginisialisasi papan catur dengan nilai awal 0.

Loop Pertama (Baris):
Loop ini akan mengakses setiap baris pada papan catur.
Dimulai dari `row = 0`, loop ini akan berjalan hingga `row < size`.
Setiap iterasi dari loop ini akan mengakses satu baris pada papan catur.
Loop Kedua (Kolom):
Di dalam loop pertama, terdapat loop kedua yang bertanggung jawab untuk mengakses setiap kolom pada papan catur.
Dimulai dari `col = 0`, loop ini akan berjalan hingga `col < size`.
Setiap iterasi dari loop ini akan mengakses satu kolom pada papan catur.

Inisialisasi Nilai 0:
Di dalam loop kedua, setiap sel pada papan catur diinisialisasi dengan nilai 0.
Ini dilakukan dengan menyimpan nilai 0 ke dalam setiap sel menggunakan pointer `chessBoard`.
Pointer `chessBoard` digunakan untuk mengakses setiap sel pada papan catur secara linear dengan menggunakan indeks baris dan kolom.

Dengan demikian, loop bersarang ini memastikan bahwa setiap sel pada papan catur diberi nilai awal 0 sebelum langkah-langkah berikutnya dieksekusi. Ini adalah langkah persiapan sebelum menandai posisi-posisi yang dapat dicapai oleh kuda.

2.

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

    for (int k = 0; k < 8; k++) {
        int new_i = i + dx[k];
        int new_j = j + dy[k];

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

Pada bagian ini, kita memiliki dua array, `dx` dan `dy`, yang menyimpan nilai perubahan posisi untuk langkah-langkah yang mungkin dilakukan oleh kuda. Setiap elemen array tersebut adalah perubahan baris dan kolom yang diperlukan untuk langkah kuda tertentu.

Kemudian, terdapat sebuah loop `for` yang berjalan sebanyak 8 kali, mengiterasi melalui semua kemungkinan langkah kuda.

Di dalam loop, dilakukan perhitungan untuk mendapatkan posisi baru kuda setelah melakukan langkah tertentu. Hal ini dilakukan dengan menambahkan perubahan posisi (`dx[k]` dan `dy[k]`) ke posisi awal kuda (`i` dan `j`).

Setelah mendapatkan posisi baru kuda, dilakukan pengecekan apakah posisi tersebut masih berada dalam batas papan catur yang valid. Jika i dan j yang baru berada dalam rentang antara 0 dan `size - 1`, maka posisi tersebut berada dalam papan catur yang valid.

Jika posisi tersebut valid, maka nilai sel pada papan catur yang sesuai dengan posisi tersebut diubah menjadi 1. Ini menandakan bahwa kuda dapat mencapai posisi tersebut dalam satu langkah.

Dengan demikian, setelah loop ini dieksekusi, semua posisi yang dapat dicapai oleh kuda dalam satu langkah akan ditandai dengan nilai 1 pada papan catur.

3.

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

Pada bagian ini, terdapat dua loop bersarang yang bertujuan untuk mencetak nilai papan catur yang telah dimodifikasi.

Loop Pertama (Baris):
Loop ini akan mengakses setiap baris pada papan catur.
Dimulai dari `row = 0`, loop ini akan berjalan hingga `row < size`.
Setiap iterasi dari loop ini akan mengakses satu baris pada papan catur.
Loop Kedua (Kolom):
Di dalam loop pertama, terdapat loop kedua yang bertanggung jawab untuk mengakses setiap kolom pada papan catur.
Dimulai dari `col = 0`, loop ini akan berjalan hingga `col < size`.
Setiap iterasi dari loop ini akan mengakses satu kolom pada papan catur.
Mencetak Nilai Sel:
Di dalam loop kedua, nilai dari setiap sel pada papan catur dicetak menggunakan `printf()`.
Nilai setiap sel diakses menggunakan pointer `chessBoard`, dengan mengalikan `row` dengan `size` dan menambahkan `col`.
Setelah mencetak semua nilai dalam satu baris, `printf("\n")` digunakan untuk mencetak baris baru.

Dengan demikian, setelah kedua loop selesai dieksekusi, seluruh nilai pada papan catur yang telah dimodifikasi akan dicetak ke layar, membentuk representasi visual dari posisi-posisi yang dapat dicapai oleh kuda dalam satu langkah.

4.

```c
int main() {
    int i, j;
    printf("Masukkan nilai i dan j dipisahkan spasi: ");
    scanf("%d %d", &i, &j);

    int chessBoard[8][8];
    koboImaginaryChess(i, j, 8, (int *)chessBoard);

    return 0;
}
```

Pada bagian ini, kita memiliki fungsi `main()` yang merupakan fungsi utama dari program.

Deklarasi Variabel:
Variabel `i` dan `j` dideklarasikan untuk menyimpan posisi awal kuda.

Meminta Input dari Pengguna:
Pesan `"Masukkan nilai i dan j dipisahkan spasi: "` dicetak ke layar menggunakan `printf()` untuk meminta pengguna memasukkan nilai `i` dan `j`.
Input yang dimasukkan oleh pengguna kemudian disimpan dalam variabel `i` dan `j` menggunakan `scanf()`.

Inisialisasi Papan Catur:
Sebuah array dua dimensi `chessBoard` dengan ukuran 8x8 dideklarasikan untuk menyimpan papan catur.

Memanggil Fungsi `koboImaginaryChess`:
Fungsi `koboImaginaryChess` dipanggil dengan argumen `i`, `j`, `8` (ukuran papan catur), dan alamat array `chessBoard`.
Perhatikan bahwa alamat array `chessBoard` dilewatkan sebagai `(int *)chessBoard` karena fungsi `koboImaginaryChess` menerima parameter pointer `int *chessBoard`.

Mengembalikan Nilai 0:
Fungsi `main()` mengembalikan nilai 0 untuk menandakan bahwa program telah berjalan dengan sukses.

Dengan demikian, pada bagian ini, pengguna diminta untuk memasukkan posisi awal kuda, kemudian papan catur akan dibuat dan diisi dengan posisi-posisi yang dapat dicapai oleh kuda dalam satu langkah.