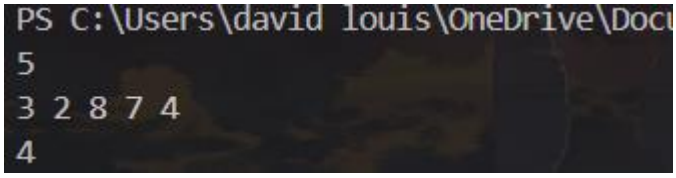
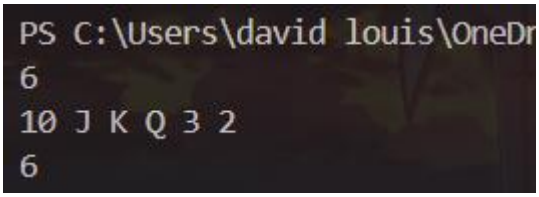
l Latihan Praktikum Algoritma Struktur Data - Array, Pointer, dan Fungsi

SOAL 1 :

```c
#include <stdio.h>

// Deklarasi fungsi
int countSwaps(int arr[], int n);
void swap(int *a, int *b);

int main() {
  // Input jumlah kartu
  int n;
  scanf("%d", &n);

  // Deklarasi array untuk menyimpan nilai kartu
  int arr[n];

  // Input nilai kartu
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }

  // Hitung jumlah minimal langkah pertukaran
  int swaps = countSwaps(arr, n);

  // Tampilkan hasil
  printf("%d\n", swaps);

  return 0;
}

// Fungsi untuk menghitung jumlah minimal langkah pertukaran
int countSwaps(int arr[], int n) {
  int swaps = 0;
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] > arr[j]) {
        swaps++;
      }
    }
  }
  return swaps;
}

// Fungsi untuk menukar dua nilai
void swap(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}

```

OUTPUT :





PENJELASAN :

Deklarasi Fungsi :

countSwaps(int arr[], int n): Menghitung jumlah minimal langkah pertukaran.

swap(int *a, int *b): Menukar dua nilai.

Fungsi **main** :

scanf("%d", &n) membaca input jumlah kartu (n) dari pengguna.

int arr[n] mendeklarasikan array arr dengan n elemen untuk menyimpan nilai kartu.

Perulangan for membaca nilai setiap kartu dan disimpan dalam array arr.

Fungsi countSwaps(arr, n) dipanggil untuk menghitung jumlah swaps.

Fungsi **countSwaps**:

Variabel swaps diinisialisasi dengan nilai 0.

Perulangan for pertama iterates dari elemen pertama (i = 0) hingga elemen kedua dari belakang (i < n - 1).

Perulangan for kedua iterates dari elemen setelah elemen ke-i (j = i + 1) hingga elemen terakhir (j < n).

Di dalam perulangan, elemen arr[i] dibandingkan dengan arr[j].

Jika arr[i] lebih besar dari arr[j], maka swaps ditambah 1.

Jumlah swaps (swaps) dikembalikan.

Fungsi **swap** :

Fungsi ini menukar dua nilai yang diwakili oleh pointer a dan b.

Penyimpanan nilai sementara: Nilai a disimpan dalam variabel temp.

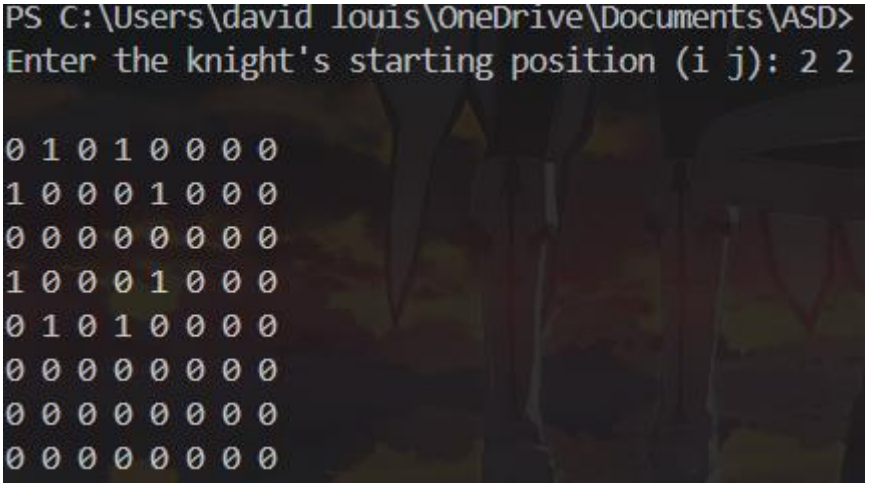
Penukaran nilai: Nilai a dan b ditukar.

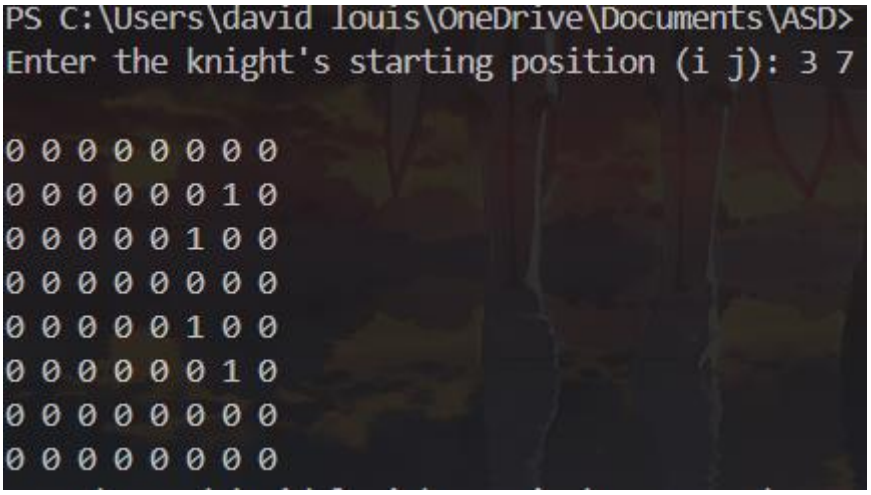
Pemulihan nilai: Nilai temp disimpan ke b.

SOAL 2 :

```c
#include <stdio.h>

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
  int possibleMoves[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 newI = i + possibleMoves[k][0];
    int newJ = j + possibleMoves[k][1];

    if (0 <= newI && newI < size && 0 <= newJ && newJ < size) {
      chessBoard[newI * size + newJ] = 1;
    }
  }
}

int main() {
  int size = 8;
  int chessBoard[size * size];

  for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
      chessBoard[i * size + j] = 0;
    }
  }

  int i, j;
  printf("Enter the knight's starting position (i j): ");
  scanf("%d %d", &i, &j);

  koboImaginaryChess(i, j, size, chessBoard);

  printf("\n");
  for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
      printf("%d ", chessBoard[i * size + j]);
    }
    printf("\n");
  }

  return 0;
}

```

OUTPUT :





PENJELASAN :

Fungsi **koboImaginaryChess :**

Fungsi ini butuh 4 info yang dikirim (parameter):

i: koordinat baris awal Kuda (bayangkan sumbu Y).

j: koordinat kolom awal Kuda (bayangkan sumbu X).

size: ukuran papan catur (misal 8x8).

chessBoard: ini pointer ke array yang nanti nyimpen informasi posisi Kuda di papan catur (nilai 1 berarti ada Kuda, 0 berarti kosong).

possibleMoves: Array dua dimensi ini nyimpen delapan kemungkinan arah gerak Kuda.

Setiap elemen array possibleMoves[i] terdiri dari dua nilai:
possibleMoves[i][0]: perubahan koordinat baris akibat gerakan (misal 2 ke atas, -2 ke bawah).
possibleMoves[i][1]: perubahan koordinat kolom akibat gerakan (misal 1 ke kanan, -1 ke kiri).

Perulangan **for** ini jalan sebanyak 8 kali (sesuai jumlah kemungkinan gerak). Di setiap iterasi:

newI dan **newJ** ngitung koordinat possible (möglichen) akibat gerakan Kuda saat ini **(possibleMoves[k][0]** dan **possibleMoves[k][1]).**

Kondisi **if** ini ngecek apakah koordinat possible (**newI** dan **newJ**) masih di dalam batas papan catur (0 sampai **size-1**). Kalo iya, baru lanjut.

Kalo koordinat possible masih di papan**, chessBoard[newI * size + newJ]** diisi nilai 1. Ini tandanya di koordinat tersebut ada Kuda. Proses ini ngisi array **chessBoard** sesuai dengan semua kemungkinan gerak Kuda.

Fungsi **main :**

size dan **chessBoard**: Kita set ukuran papan catur (**size = 8**) dan deklarasi array **chessBoard** yang cukup buat nyimpen informasi seluruh kotak (ingat **size * size**).

Inisialisasi Papan Catur: Looping **for** ini ngisi semua elemen **chessBoard** dengan nilai 0, yang artinya seluruh kotak awalnya kosong (belum ada Kuda).

Input Posisi Awal Kuda: Baris **printf** minta kita nginput koordinat awal Kuda (baris dan kolom) yang disimpan di variabel **i** dan **j**.

Simulasi Gerakan: Kita panggil fungsi **koboImaginaryChess** dengan parameter awal Kuda **(i, j)**, ukuran papan (**size**), dan pointer ke **chessBoard**. Fungsi ini kemudian menghitung dan menandai semua kemungkinan gerak Kuda di array **chessBoard**.

Print Papan Catur: Looping **for** ini ngeprint isi array **chessBoard** ke layar. Nilai 1 menunjukkan ada Kuda, dan 0 menunjukkan kotak kosong. Ini ngasih gambaran semua kemungkinan gerak Kuda dari posisi awal yang kita input.