Nama : Gede Satyamahinsa Prastita Uttama

NIM : 1203220054

Kelas : IF-02-03

Latihan 2 Praktikum Algoritma Struktur Data – Fungsi & Pointer

1. Source Code:

```c
#include <stdio.h>

// greatestOf() => digunakan untuk mencari nilai maksimum dari 4 bilangan yang ada di dalam parameter.
int greatestOf(int a, int b, int c, int d) {
  // Inisialisasi variabel max bertipe data integer dengan nilai, yaitu a.
  int max = a;
  // Melakukan pengecekan menggunakan if secara bergantian antara
  // variabel max dengan bilangan selanjutnya yang ada di dalam parameter, yaitu b, c, dan d.
  // Pengecekan secara berurutan ini dilakukan agar semua bilangan
  // melewati proses perbandingan sehingga dapat menentukan nilai maksimum dari 4 bilangan.
  if(b > max) {
    max = b;
  }
  if(c > max) {
    max = c;
  }
  if(d > max) {
    max = d;
  }

  // Mengembalikan nilai max.
  return max;
}

int main() {
  // Deklarasi 4 variabel bertipe data integer.
  int a, b, c, d;
  // User melakukan input nilai ke dalam 4 variabel tersebut.
  scanf("%d %d %d %d", &a, &b, &c, &d);
  // Menampilkan nilai kembali dari fungsi greatestOf(), yaitu
  // nilai maksimum dari 4 bilangan yang ada di dalam argumen.
  printf("%d", greatestOf(a, b, c, d));
  return 0;
}
```

Output:

```
4 5 3 2
5
```

```
101 27 26 998
998
```

2. Source Code:

```c
#include <stdio.h>

// arrayMean() ⇒ digunakan untuk mencari nilai rata-rata dari sebuah array yang memiliki N elemen.
float arrayMean(int *arr, int N) {
  // Inisialisasi variabel sum bertipe data float dengan nilai awal, yaitu 0.
  float sum = 0;
  // Melakukan perulangan di dalam array sebanyak N kali.
  // Perulangan ini dilakukan untuk mendapatkan elemen yang ada di dalam array.
  for(int i = 0; i < N; i++) {
    // Variabel sum ditambahkan dengan setiap elemen di dalam array berdasarkan indeks ke-i.
    sum += *(arr + i);
  }
  // Mengembalikan nilai rata-rata dari array,
  // yaitu pembagian antara hasil penjumlahan semua elemen di dalam array
  // dengan banyak elemen yang ada di dalam array.
  return sum / N;
}

int main() {
  // Deklarasi variabel N yang nantinya digunakan sebagai
  // banyak elemen yang ada di dalam array.
  int N;
  // User melakukan input nilai ke dalam variabel N.
  scanf("%d", &N);

  // Deklarasi variabel arr sebagai array yang memiliki N elemen di dalamnya.
  int arr[N];
  // Perulangan ini dilakukan untuk menampung input dari user
  // berupa angka integer sebanyak N elemen yang dipisahkan tanda spasi.
  for(int i = 0; i < N; i++) {
    // User melakukan input nilai ke dalam variabel arr dengan indeks ke-i.
    scanf("%d", &arr[i]);
  }

  // Menampilkan nilai kembali dari fungsi arrayMean() berupa
  // hasil rata-rata dari sebuah array dengan presisi hingga 2 angka di belakang koma.
  printf("%.2f", arrayMean(arr, N));
  return 0;
}
```

Output:

```
4
3 7 1 10
5.25
```

```
11
4 6 10 30 22 11 89 62 78 24 1
30.64
```

3. Source Code:

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

// koboImaginaryChess() ⇒ digunakan untuk mensimulasikan skenario yang dimiliki oleh Kobo.
// Simulasi tersebut, yaitu di dalam papan catur berukuran 8x8,
// Kobo ingin mengetahui posisi mana saja yang dapat dicapai oleh
// bidak kuda dalam sekali jalan apabila bidak tersebut berada pada posisi i, j.
// Fungsi koboImaginaryChess() memiliki 4 buah parameter, yaitu
// i sebagai posisi kuda dalam baris, j sebagai posisi kuda dalam kolom,
// size sebagai ukuran papan catur, dan
// chessBoard sebagai papan catur yang berisi elemen sebanyak 64.
void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
  // Inisialisasi sebuah array bernama posisi sebagai langkah atau jalan
  // yang mungkin dilalui oleh bidak kuda (kuda berjalan seperti membentuk huruf L).
  // Kumpulan array di dalam array posisi memiliki 2 buah elemen masing-masingnya,
  // yaitu indeks ke-0 sebagai jalan kuda yang mengacu kepada baris dan
  // indeks ke-1 sebagai jalan kuda yang mengacu kepada kolom.
  int posisi[][2] = {{-2, -1}, {-2, 1}, {-1, -2}, {-1, 2}, {2, -1}, {2, 1}, {1, -2}, {1, 2}};

  // Perulangan ini dilakukan untuk mengecek setiap kemungkinan langkah atau jalan
  // yang dapat dilalui oleh bidak kuda sesuai dengan array posisi.
  for(int k = 0; k < 8; k++) {
    // Inisialisasi variabel baris dan kolom bertipe data integer
    // dengan nilai sesuai dengan setiap array yang ada di dalam array posisi.
    int baris = posisi[k][0], kolom = posisi[k][1];
    // Melakukan pengecekan apakah setiap jalan yang dilalui oleh bidak kuda
    // melewati papan catur atau tidak.
    if(i + baris ≥ 0 && i + baris < 8 && j + kolom ≥ 0 && j + kolom < 8) {
      // Jika jalan bidak kuda tidak melewati papan catur,
      // maka nilai papan catur chessBoard dengan indeks baris (i + baris) dan
      // indeks kolom (j + kolom) diubah menjadi 1.
      // *(chessBoard + (i + baris) * 8 + (j + kolom)) sama artinya
      // dengan chessBoard[(i + baris) * 8][j + kolom]
      *(chessBoard + (i + baris) * 8 + (j + kolom)) = 1;
    }
  }
}
```

```c
// print_array() ⇒ digunakan untuk menampilkan semua elemen yang ada di dalam array chessBoard.
void print_array(int *chessBoard) {
  // Perulangan pertama berfungsi untuk melakukan pemanggilan terhadap baris.
  for(int i = 0; i < 8; i++) {
    // Perulangan kedua berfungsi untuk melakukan pemanggilan terhadap kolom.
    for(int j = 0; j < 8; j++) {
      // Menampilkan elemen yang ada di dalam array dengan indeks baris ke i*8
      // (karena setiap baris ada 8 elemen) dan indeks kolom ke j.
      // *(chessBoard + i * 8 + j) sama dengan chessBoard[i * 8][j]
      printf("%d", *(chessBoard + i * 8 + j));
    }
    // \n digunakan untuk membuat baris baru.
    printf("\n");
  }
}

int main() {
  // Deklarasi variabel i dan j bertipe data integer sebagai posisi awal bidak kuda.
  int i, j;
  // User melakukan input nilai i sebagai posisi baris dan j sebagai posisi kolom.
  scanf("%d %d", &i, &j);
  // Inisialisasi variabel size bertipe data integer sebagai ukuran papan catur.
  int size = 8;
  // Mengalokasikan memori secara dinamis sejumlah 256 byte karena
  // setiap tipe data integer membutuhkan 4 byte memori dalam menyimpan sebuah nilai.
  int *chessBoard = (int*) malloc(size * size * sizeof(int));
  // Perulangan ini dilakukan untuk membuat semua elemen di dalam array chessBoard menjadi 0.
  for (int k = 0; k < size * size; k++) {
    // *(chessBoard + k) untuk mengakses setiap elemen yang ada di dalam array chessBoard.
    *(chessBoard + k) = 0;
  }

  // Memanggil fungsi koboImaginaryChess() untuk menjalankan skenario yang diinginkan oleh Kobo.
  koboImaginaryChess(i, j, size, chessBoard);
  // Memanggil fungsi print_array() untuk menampilkan seluruh elemen
  // yang ada di dalam array chessBoard dan mengetahui posisi mana saja yang
  // dapat dicapai oleh bidak kuda.
  print_array(chessBoard);
  return 0;
}
```

Output:

```
2 2
01010000
10001000
00000000
10001000
01010000
00000000
00000000
00000000
```

```
3 7
00000000
00000010
00000100
00000000
00000100
00000010
00000000
00000000
```

```
7 4
00000000
00000000
00000000
00000000
00000000
00010100
00100010
00000000
```