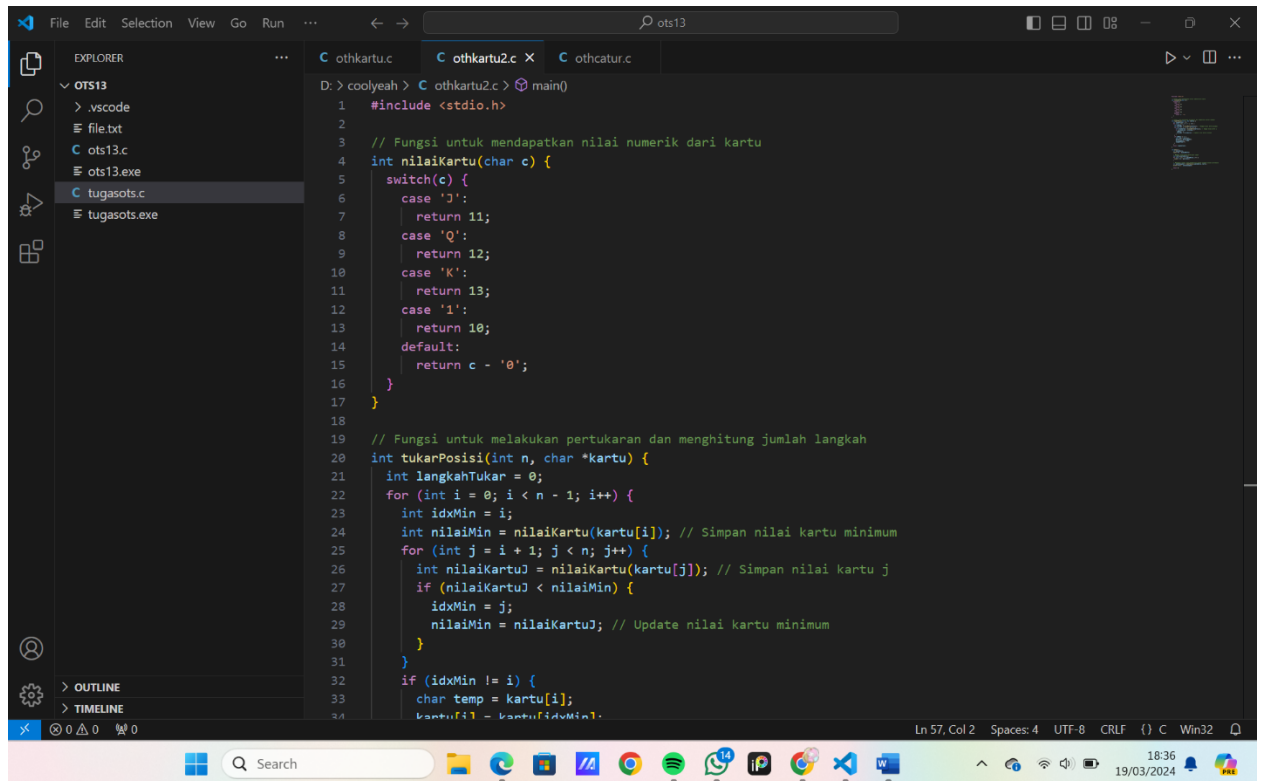


Nama : Maria Rosa Wahyuning Utami
NIM : 1203230123
Kelas : IF 03-03

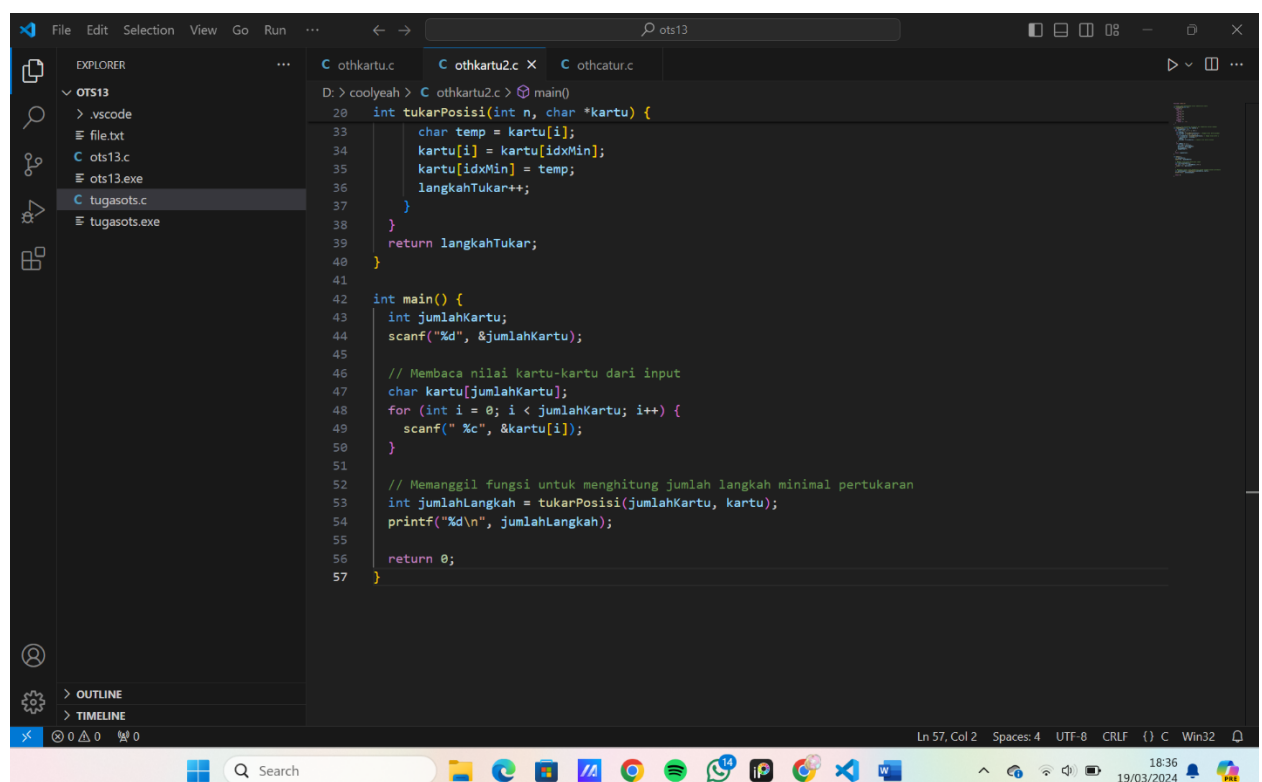
Tugas OTH Algoritma & Struktur Data Week 4

1. Kartu

a. Source Code

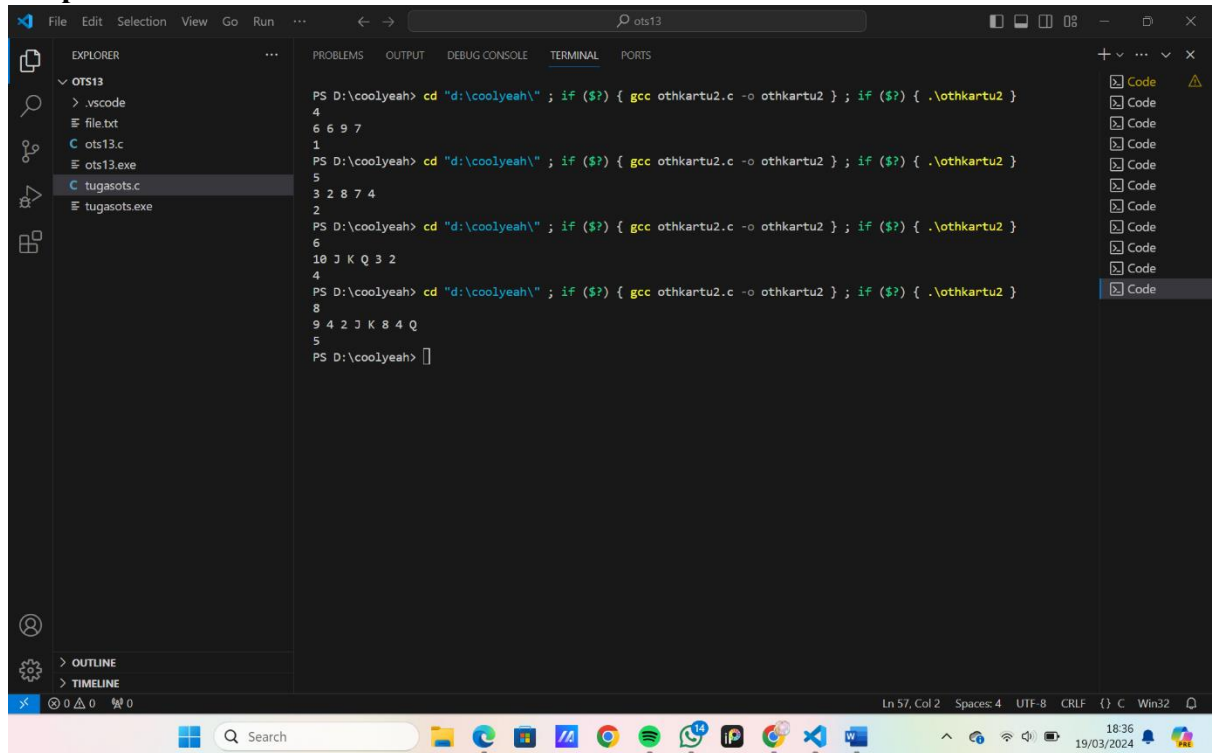


```
1  #include <stdio.h>
2
3  // Fungsi untuk mendapatkan nilai numerik dari kartu
4  int nilaiKartu(char c) {
5      switch(c) {
6          case 'J':
7              return 11;
8          case 'Q':
9              return 12;
10         case 'K':
11             return 13;
12         case '1':
13             return 10;
14         default:
15             return c - '0';
16     }
17 }
18
19 // Fungsi untuk melakukan pertukaran dan menghitung jumlah langkah
20 int tukarPosisi(int n, char *kartu) {
21     int langkahTukar = 0;
22     for (int i = 0; i < n - 1; i++) {
23         int idxMin = i;
24         int nilaiMin = nilaiKartu(kartu[i]); // Simpan nilai kartu minimum
25         for (int j = i + 1; j < n; j++) {
26             int nilaiKartuJ = nilaiKartu(kartu[j]); // Simpan nilai kartu j
27             if (nilaiKartuJ < nilaiMin) {
28                 idxMin = j;
29                 nilaiMin = nilaiKartuJ; // Update nilai kartu minimum
30             }
31         }
32         if (idxMin != i) {
33             char temp = kartu[i];
34             kartu[i] = kartu[idxMin];
35             kartu[idxMin] = temp;
36             langkahTukar++;
37         }
38     }
39     return langkahTukar;
40 }
41
42 int main() {
43     int jumlahKartu;
44     scanf("%d", &jumlahKartu);
45
46     // Membaca nilai kartu-kartu dari input
47     char kartu[jumlahKartu];
48     for (int i = 0; i < jumlahKartu; i++) {
49         scanf(" %c", &kartu[i]);
50     }
51
52     // Memanggil fungsi untuk menghitung jumlah langkah minimal pertukaran
53     int jumlahLangkah = tukarPosisi(jumlahKartu, kartu);
54     printf("%d\n", jumlahLangkah);
55
56     return 0;
57 }
```



```
20 int tukarPosisi(int n, char *kartu) {
33     char temp = kartu[i];
34     kartu[i] = kartu[idxMin];
35     kartu[idxMin] = temp;
36     langkahTukar++;
37 }
38 }
39 return langkahTukar;
40 }
41
42 int main() {
43     int jumlahKartu;
44     scanf("%d", &jumlahKartu);
45
46     // Membaca nilai kartu-kartu dari input
47     char kartu[jumlahKartu];
48     for (int i = 0; i < jumlahKartu; i++) {
49         scanf(" %c", &kartu[i]);
50     }
51
52     // Memanggil fungsi untuk menghitung jumlah langkah minimal pertukaran
53     int jumlahLangkah = tukarPosisi(jumlahKartu, kartu);
54     printf("%d\n", jumlahLangkah);
55
56     return 0;
57 }
```

b. Output



```
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc othkartu2.c -o othkartu2 } ; if ($?) { .\othkartu2 }
4
6 6 9 7
1
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc othkartu2.c -o othkartu2 } ; if ($?) { .\othkartu2 }
5
3 2 8 7 4
2
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc othkartu2.c -o othkartu2 } ; if ($?) { .\othkartu2 }
6
10 J K Q 3 2
4
PS D:\coolyeah> cd "d:\coolyeah\" ; if ($?) { gcc othkartu2.c -o othkartu2 } ; if ($?) { .\othkartu2 }
8
9 4 2 J K 8 4 Q
5
PS D:\coolyeah>
```

c. Penjelasan program

```
#include <stdio.h>
deklarasi library untuk fungsi input dan output standar

// Fungsi untuk mendapatkan nilai numerik dari kartu
int nilaiKartu(char c) {
    deklarasi fungsi 'nilaiKartu' yang menerima karakter tunggal
    ('char c') dan mengembalikan nilai numerik dari kartu
    tersebut. fungsi ini akan digunakan untuk mengonversi
    karakter kartu menjadi nilai numerik

    switch(c) {
        awal dari struktur switch-case yang akan mengevaluasi nilai
        karakter c

        case 'J':
            return 11;
            jika karakter c adalah 'J', fungsi akan mengembalikan nilai
            11

        case 'Q':
            return 12;
            jika karakter c adalah 'Q', fungsi akan mengembalikan nilai
            12

        case 'K':
            return 13;
```

jika karakter c adalah 'K', fungsi akan mengembalikan nilai 13

```
case '1':
```

```
return 10;
```

jika karakter c adalah '1', fungsi akan mengembalikan nilai 10

```
default:
```

```
return c - '0';
```

jika karakter c bukan 'J', 'K', 'Q', atau '1' maka fungsi akan mengembalikan nilai numerik dari karakter tersebut dengan mengurangi karakter '0'. ini bekerja karena dalam representasi ASCII (standar yang mengaitkan setiap karakter dengan sebuah bilangan integer), karakter '0' memiliki nilai numerik 48, '1' memiliki nilai 49, dan seterusnya, sehingga pengurangan '0' menghasilkan nilai numerik yang diinginkan

```
}
```

```
}
```

akhir dari switch-case dan fungsi nilaiKartu

```
// Fungsi untuk melakukan pertukaran dan menghitung jumlah langkah
```

```
int tukarPosisi(int n, char *kartu) {
```

deklarasi fungsi bernama tukarPosisi yang mengambil dua argument, sebuah integer n dan sebuah array karakter kartu

```
int langkahTukar = 0;
```

variabel langkahTukar digunakan untuk menghitung jumlah langkah pertukaran posisi yang dilakukan

```
for (int i = 0; i < n - 1; i++) {
```

loop for akan berjalan dari 0 hingga n-2 (indeks terakhir sebelum n-1), mengiterasi melalui kartu-kartu dalam array

```
int idxMin = i;
```

```
int nilaiMin = nilaiKartu(kartu[i]); // Simpan nilai kartu minimum
```

variabel idxMin akan menyimpan indeks kartu dengan nilai minimum dalam sisa array. variabel nilaiMin akan menyimpan nilai numerik dari kartu tersebut

```
for (int j = i + 1; j < n; j++) {
```

loop for nested akan berjalan dari indeks setelah i hingga n-1 untuk memeriksa kartu-kartu yang tersisa setelah kartu dengan indeks i

```

int nilaiKartuJ = nilaiKartu(kartu[j]); // Simpan nilai kartu
j
variabel nilaiKartuJ menyimpan nilai numerik dari kartu yang
sedang diperiksa

if (nilaiKartuJ < nilaiMin) {
jika nilai kartu yang sedang diperiksa kurang dari nilai
minimum yang telah disimpan sebelumnya,

    idxMin = j;
    nilaiMin = nilaiKartuJ; // Update nilai kartu minimum
    indeks minimum dan nilai minimum diperbarui dengan nilai dari
    kartu yang sedang diperiksa

}
}
akhir dari loop for nested

if (idxMin != i) {
setelah menemukan kartu dengan nilai minimum, jika indeks
minimum tidak sama dengan indeks i, berarti kartu dengan nilai
minimum tersebut tidak berada di posisi yang benar

    char temp = kartu[i];
    kartu[i] = kartu[idxMin];
    kartu[idxMin] = temp;
    kartu pada posisi i dan idxMin ditukar posisinya menggunakan
    variabel temp

    langkahTukar++;
    jumlah langkah pertukaran posisi ditingkatkan

}
}
akhir dari loop for

return langkahTukar;
fungsi mengembalikan jumlah Langkah pertukaran posisi yang
dilakukan

}
akhir dari fungsi tukarPosisi

int main() {
ini adalah fungsi utama dari program

```

```
int jumlahKartu;
```

variabel jumlahKartu digunakan untuk menyimpan jumlah kartu yang akan diinput oleh user

```
scanf("%d", &jumlahKartu);
```

scanf() digunakan untuk membaca jumlah tersebut dari input standar

```
// Membaca nilai kartu-kartu dari input
```

```
char kartu[jumlahKartu];
```

deklarasi membuat array 'kartu' yang memiliki Panjang sejumlah 'jumlahKartu'. setiap elemen array ini akan menampung satu karakter, mewakili satu kartu

```
for (int i = 0; i < jumlahKartu; i++) {
```

inisialisasi loop for dimulai dari 'i = 0' dan akan berjalan selama 'i' kurang dari 'jumlahKartu'. loop ini membaca setiap kartu satu per satu

```
scanf(" %c", &kartu[i]);
```

```
}
```

fungsi 'scanf' digunakan untuk membaca inputan user. format string " %c" digunakan untuk membaca satu karakter. spasi sebelum '%c' digunakan untuk mengabaikan karakter whitespace (termasuk spasi, tab, newline) sebelum karakter yang sebenarnya ingin dibaca. karakter yang dibaca kemudian disimpan dalam elemen array 'kartu' pada indeks 'i'

```
// Memanggil fungsi untuk menghitung jumlah langkah minimal pertukaran
```

```
int jumlahLangkah = tukarPosisi(jumlahKartu, kartu);
```

fungsi tukarPosisi() dipanggil untuk menghitung jumlah langkah minimal yang diperlukan untuk memindahkan kartu-kartu ke posisi yang terurut

```
printf("%d\n", jumlahLangkah);
```

hasil jumlah langkah minimal yang diperlukan untuk memindahkan kartu-kartu ke posisi yang terurut akan dicetak

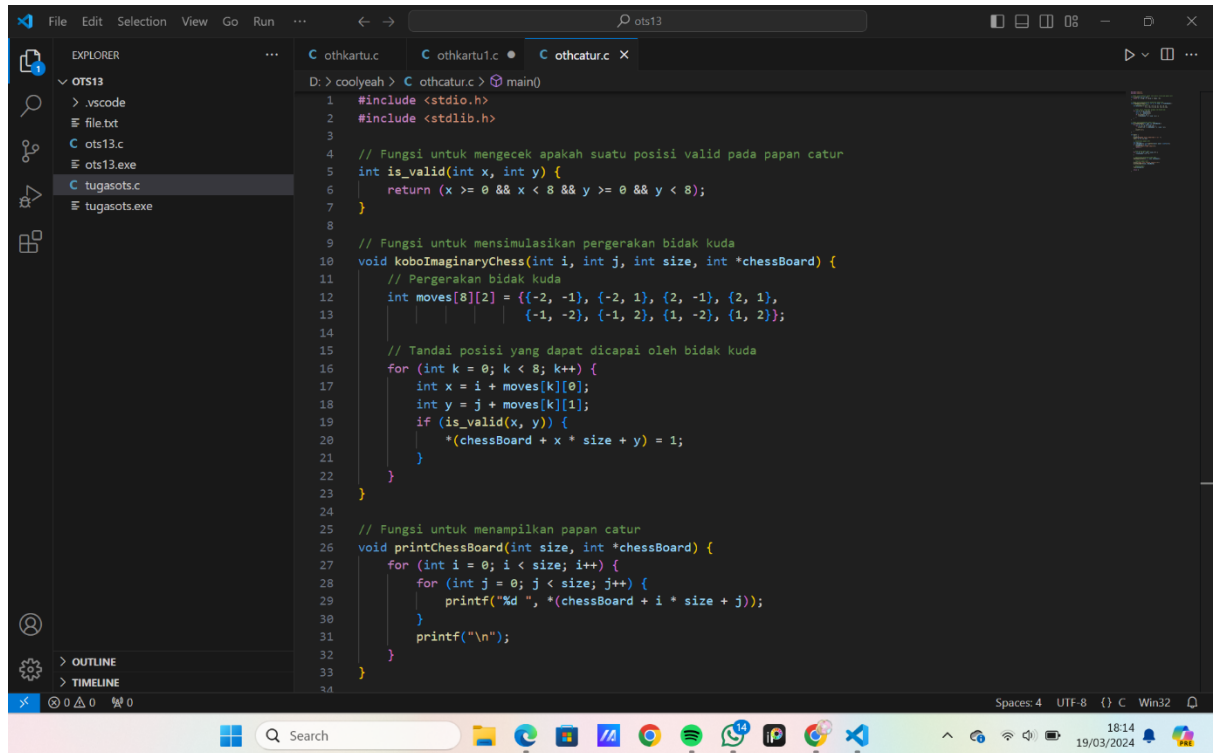
```
return 0;
```

```
}
```

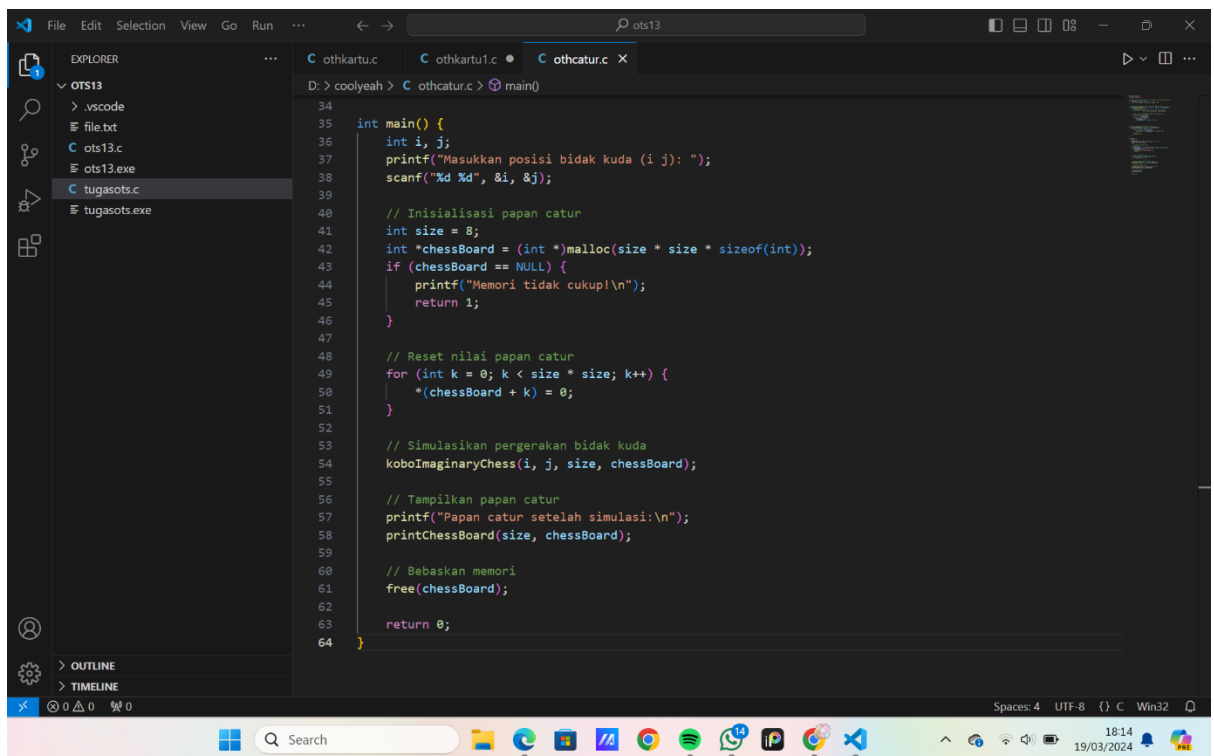
akhir dari fungsi main, dan memberi tahu sistem operasi bahwa program telah selesai dijalankan dengan sukses. nilai '0' yang dikembalikan oleh 'return 0;' menandakan bahwa program telah berjalan dengan baik dan tidak ada kesalahan yang terjadi

2. Catur

a. Source Code

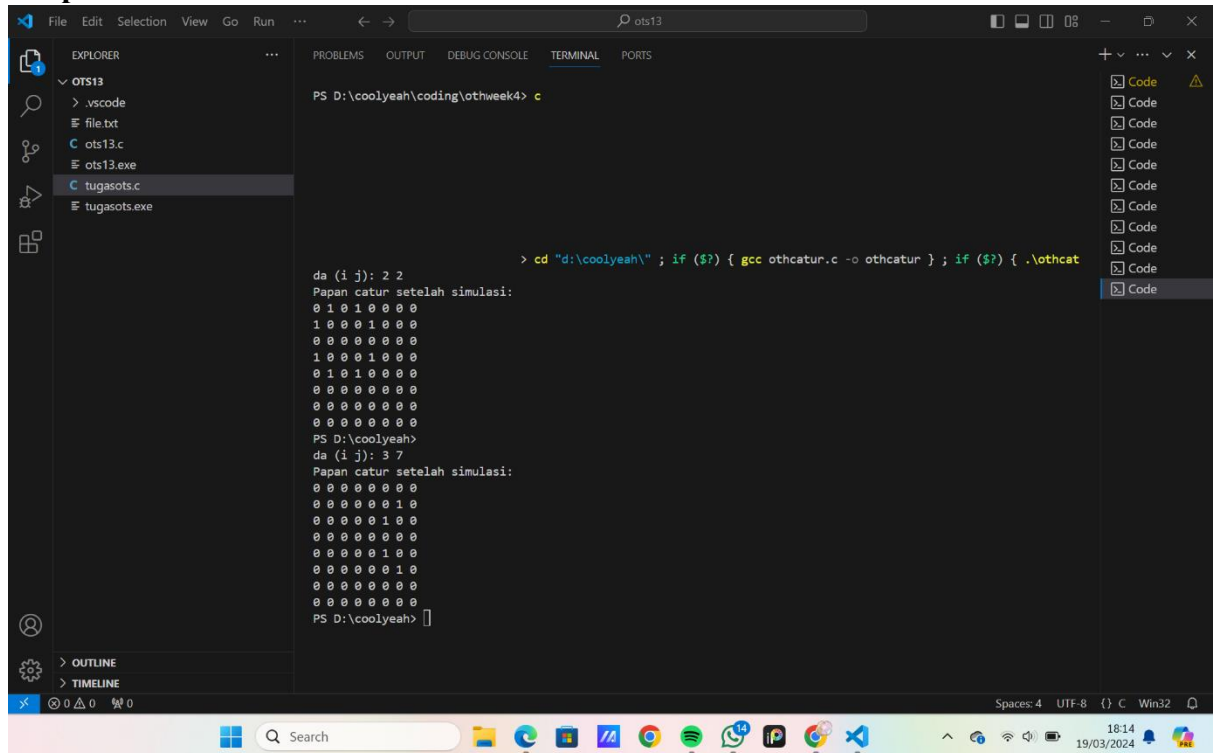


```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Fungsi untuk mengecek apakah suatu posisi valid pada papan catur
5 int is_valid(int x, int y) {
6     return (x >= 0 && x < 8 && y >= 0 && y < 8);
7 }
8
9 // Fungsi untuk mensimulasikan pergerakan bidak kuda
10 void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
11     // Pergerakan bidak kuda
12     int moves[8][2] = {{-2, -1}, {-2, 1}, {2, -1}, {2, 1},
13                       {-1, -2}, {-1, 2}, {1, -2}, {1, 2}};
14
15     // Tandai posisi yang dapat dicapai oleh bidak kuda
16     for (int k = 0; k < 8; k++) {
17         int x = i + moves[k][0];
18         int y = j + moves[k][1];
19         if (is_valid(x, y)) {
20             *(chessBoard + x * size + y) = 1;
21         }
22     }
23 }
24
25 // Fungsi untuk menampilkan papan catur
26 void printChessBoard(int size, int *chessBoard) {
27     for (int i = 0; i < size; i++) {
28         for (int j = 0; j < size; j++) {
29             printf("%d ", *(chessBoard + i * size + j));
30         }
31         printf("\n");
32     }
33 }
34
```



```
34
35 int main() {
36     int i, j;
37     printf("Masukkan posisi bidak kuda (i j): ");
38     scanf("%d %d", &i, &j);
39
40     // Inisialisasi papan catur
41     int size = 8;
42     int *chessBoard = (int *)malloc(size * size * sizeof(int));
43     if (chessBoard == NULL) {
44         printf("Memori tidak cukup!\n");
45         return 1;
46     }
47
48     // Reset nilai papan catur
49     for (int k = 0; k < size * size; k++) {
50         *(chessBoard + k) = 0;
51     }
52
53     // Simulasikan pergerakan bidak kuda
54     koboImaginaryChess(i, j, size, chessBoard);
55
56     // Tampilkan papan catur
57     printf("Papan catur setelah simulasi:\n");
58     printChessBoard(size, chessBoard);
59
60     // Bebaskan memori
61     free(chessBoard);
62
63     return 0;
64 }
```

b. Output



```
PS D:\coolyeah\coding\othweek4> c
> cd "d:\coolyeah\" ; if ($?) { gcc othcat.c -o othcat } ; if ($?) { .\othcat
da (i j): 2 2
Papan catur setelah simulasi:
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:\coolyeah>
da (i j): 3 7
Papan catur setelah simulasi:
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:\coolyeah>
```

c. Penjelasan program

```
#include <stdio.h>
untuk fungsi input-output standar seperti 'printf' dan
'scanf'

#include <stdlib.h>
diperlukan untuk fungsi 'malloc' dan 'free'

// Fungsi untuk mengecek apakah suatu posisi valid pada papan
catur
int is_valid(int x, int y) {
    deklarasi fungsi 'is_valid' yang mengambil dua argumen
    integer 'x' dan 'y' yang mewakili posisi pada papan catur.
    fungsi ini akan mengembalikan nilai integer yang menunjukkan
    apakah posisi yang diberikan valid atau tidak

    return (x >= 0 && x < 8 && y >= 0 && y < 8);
}
fungsi ini menggunakan ekspresi logika untuk mengevaluasi
apakah 'x' berada dalam rentang 0 hingga 7 dan 'y' berada
dalam rentang yang sama. jika posisi tersebut valid, fungsi
akan mengembalikan nilai '1', jika tidak, akan mengembalikan
nilai '0'

// Fungsi untuk mensimulasikan pergerakan bidak kuda
void koboImaginaryChess(int i, int j, int size, int
*chessBoard) {
```

deklarasi fungsi 'koboImaginaryChess' yang mengambil empat argumen: dua integer 'i' dan 'j' mewakili posisi awal bidak kuda pada papan catur, integer 'size' yang menunjukkan ukuran papan catur (8x8), dan pointer ke array integer 'chessBoard' yang merepresentasikan papan catur

```
// Pergerakan bidak kuda
```

```
int moves[8][2] = {{-2, -1}, {-2, 1}, {2, -1}, {2, 1},  
                  {-1, -2}, {-1, 2}, {1, -2}, {1, 2}};
```

array 'moves' menyimpan langkah-langkah yang mungkin dilakukan oleh bidak kuda. setiap elemen array adalah array dua dimensi yang menyimpan perubahan posisi 'x' dan 'y' untuk masing-masing langkah

```
// Tandai posisi yang dapat dicapai oleh bidak kuda
```

```
for (int k = 0; k < 8; k++) {  
loop 'for' digunakan untuk mengiterasi melalui setiap kemungkinan langkah yang dapat dilakukan oleh bidak kuda. terdapat 8 kemungkinan langkah yang disimpan dalam array 'moves'
```

```
int x = i + moves[k][0];
```

```
int y = j + moves[k][1];
```

variabel 'x' dan 'y' digunakan untuk menyimpan koordinat posisi yang akan ditandai. koordinat ini dihitung dengan menambahkan perubahan posisi dari array 'moves' ke koordinat awal bidak kuda ('i' dan 'j')

```
if (is_valid(x, y)) {
```

pemeriksaan dilakukan untuk memastikan apakah posisi yang dihasilkan ('x' dan 'y') valid atau tidak. fungsi 'is_valid' dipanggil untuk memeriksa validitas posisi tersebut

```
*(chessBoard + x * size + y) = 1;
```

jika posisi 'x' dan 'y' valid, nilai papan catur pada posisi tersebut ditandai dengan '1'. hal ini dilakukan dengan mengakses elemen array 'chessBoard' pada indeks yang sesuai dengan koordinat posisi ('x', 'y') dan mengubah nilainya menjadi '1'. ini menunjukkan bahwa posisi tersebut dapat dicapai oleh bidak kuda

```
}
```

```
}
```

```
}
```

```
// Fungsi untuk menampilkan papan catur
```

```
void printChessBoard(int size, int *chessBoard) {
```

deklarasi fungsi 'printChessBoard' yang bertujuan untuk menampilkan papan catur ke layar. fungsi ini menerima dua

argumen: integer 'size' yang merupakan ukuran papan catur dan pointer ke array integer 'chessBoard' yang merepresentasikan papan catur

```
for (int i = 0; i < size; i++) {
```

loop for ini akan iterasi melalui setiap baris pada papan catur. variabel 'i' digunakan untuk menyimpan indeks baris saat ini

```
for (int j = 0; j < size; j++) {
```

loop for bersarang ini akan iterasi melalui setiap kolom pada papan catur untuk baris saat ini. variabel 'j' digunakan untuk menyimpan indeks kolom saat ini

```
printf("%d ", *(chessBoard + i * size + j));
```

mencetak nilai yang disimpan di posisi '(i, j)' pada papan catur. Notasi '*(chessboard + i * size + j)' digunakan untuk mengakses nilai di posisi '(i, j)' dalam array dua dimensi 'chessBoard'. variabel 'size' digunakan untuk menentukan jumlah kolom dalam satu baris

```
    }  
    printf("\n");  
}
```

setelah loop dalam baris saat ini selesai, 'printf("\n")' digunakan untuk mencetak karakter baris baru, sehingga setiap baris dari papan catur dipisahkan oleh baris baru

```
int main() {
```

memulai fungsi utama dari program

```
int i, j;
```

deklarasi membuat dua variabel 'i' dan 'j' yang akan digunakan untuk menyimpan posisi bidak kuda yang diinput oleh user

```
printf("Masukkan posisi bidak kuda (i j): ");
```

perintah untuk mencetak pesan ke layar yang meminta user untuk menginput posisi bidak kuda

```
scanf("%d %d", &i, &j);
```

fungsi 'scanf' digunakan untuk membaca masukan dari pengguna. Format string "%d %d" mengindikasikan bahwa program akan membaca dua integer yang dipisahkan oleh spasi atau karakter whitespace. nilai yang dibaca akan disimpan ke dalam variabel 'i' dan 'j'

```
// Inisialisasi papan catur
int size = 8;
deklarasi variabel 'size' dan inisialisasinya dengan nilai 8.
variabel ini menunjukkan ukuran papan catur, yang dalam hal
ini adalah papan catur berukuran 8x8

int *chessBoard = (int *)malloc(size * size * sizeof(int));
deklarasi variabel pointer 'chessBoard' yang akan menunjuk ke
blok memori yang dialokasikan untuk menyimpan papan catur.
fungsi 'malloc' digunakan untuk mengalokasikan memori sebesar
'size * size * sizeof(int)' byte, yang sesuai dengan jumlah
sel pada papan catur. hasilnya disimpan dalam pointer
'chessBoard'. penyesuaian tipe menggunakan '(int *)' karena
'malloc' mengembalikan pointer ke blok memori tanpa tipe

if (chessBoard == NULL) {
printf("Memori tidak cukup!\n");
return 1;
}
pengecekan apakah alokasi memori berhasil dilakukan. Jika
'chessBoard' sama dengan 'NULL', maka alokasi memori gagal
karena memori yang tersedia tidak mencukupi. program mencetak
pesan kesalahan dan mengembalikan nilai 1 sebagai tanda bahwa
program mengalami kegagalan dan harus dihentikan

// Reset nilai papan catur
for (int k = 0; k < size * size; k++) {
loop for ini akan iterasi melalui setiap sel pada papan catur.
variabel 'k' digunakan untuk menyimpan indeks sel saat ini,
dimulai dari 0 hingga '(size * size) - 1'

*(chessBoard + k) = 0;
}
pernyataan ini mengatur nilai sel pada indeks 'k' di papan
catur menjadi '0'. notasi '*(chessBoard + k)' digunakan untuk
mengakses nilai sel pada posisi 'k' dalam array satu dimensi
'chessBoard'. nilai 0 menandakan bahwa tidak ada bidak yang
berada dalam posisi tersebut

// Simulasikan pergerakan bidak kuda
koboImaginaryChess(i, j, size, chessBoard);
fungsi 'koboImaginaryChess' dipanggil untuk mensimulasikan
pergerakan bidak kuda dari posisi awal yang diinput oleh user

// Tampilkan papan catur
printf("Papan catur setelah simulasi:\n");
printChessBoard(size, chessBoard);
```

setelah simulasi pergerakan bidak kuda selesai, papan catur ditampilkan menggunakan fungsi 'printChessBoard'

```
// Bebaskan memori
```

```
free(chessBoard);
```

memori yang dialokasikan untuk papan catur dibebaskan kembali menggunakan fungsi 'free' agar memori tersebut dapat digunakan kembali untuk penggunaan lain dan untuk mencegah kebocoran memori

```
return 0;
```

```
}
```

akhir dari fungsi 'main'. Nilai Kembali '0' menandakan bahwa program telah berakhir dengan sukses