Komponen Penilaian	Ya	Tidak
Soal 1 sesuai dengan output yang diinginkan	✓	
Soal 2 sesuai dengan output yang diinginkan	✓	
Bonus soal 1 dikerjakan	✓	

No 1

Source Code

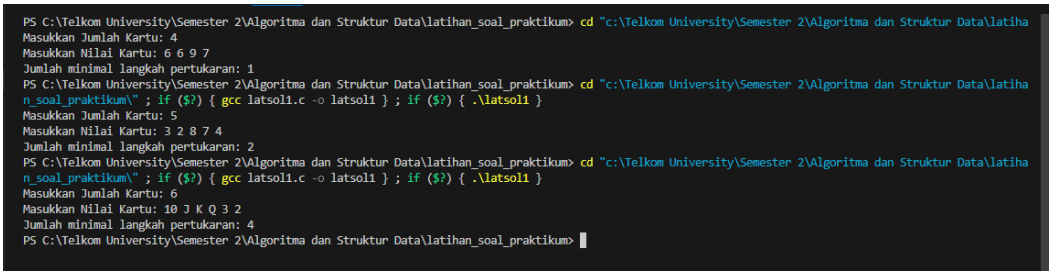
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Fungsi untuk menghitung jumlah langkah pertukaran yang diperlukan
int countSwaps(int *cards, int n) {
    int swaps = 0;

    // Menggunakan algoritma Selection Sort untuk mengurutkan kartu
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (cards[j] < cards[minIndex]) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            // Jika kartu dengan nilai terkecil tidak berada di posisi saat ini, lakukan pertukaran
            int temp = cards[i];
```

```
            cards[i] = cards[minIndex];
            cards[minIndex] = temp;
            swaps++;
        }
    }

    return swaps;
}

int main() {
    int n;
    // Meminta input jumlah kartu
    printf("Masukkan Jumlah Kartu: ");
    if (scanf("%d", &n) != 1) {
        printf("Invalid input\n");
        return 1;
    }

    int *cards = (int *)malloc(n * sizeof(int));
    if (cards == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Membaca nilai kartu
```

```
printf("Masukkan Nilai Kartu: ");
for (int i = 0; i < n; i++) {
    char value[10];
    if (scanf("%s", value) != 1) {
        printf("Invalid input\n");
        free(cards);
        return 1;
    }
    if (strcmp(value, "J") == 0) {
        cards[i] = 11;
    } else if (strcmp(value, "Q") == 0) {
        cards[i] = 12;
    } else if (strcmp(value, "K") == 0) {
        cards[i] = 13;
    } else if (strcmp(value, "A") == 0) {
        cards[i] = 1;
    } else {
        cards[i] = atoi(value);
        if (cards[i] < 1 || cards[i] > 10) {
            printf("Invalid input\n");
            free(cards);
            return 1;
        }
    }
}
```

```
    // Menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu
    int minSwaps = countSwaps(cards, n);

    // Menampilkan output sesuai yang diharapkan
    printf("Jumlah minimal langkah pertukaran: %d\n", minSwaps);

    free(cards); // Dealokasi memori
    return 0;
}
```

SS Output



Penjelasan

1. Fungsi countSwaps:
   - Fungsi ini menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu menggunakan algoritma Selection Sort.
   - Fungsi ini menerima dua parameter: cards (array yang berisi nilai-nilai kartu) dan n (jumlah kartu).
   - Fungsi ini mengembalikan jumlah langkah pertukaran yang diperlukan.

2. Fungsi main:
   - Fungsi utama program.
   - Pertama, program meminta input jumlah kartu dari pengguna.
   - Kemudian, program mengalokasikan memori untuk array cards dengan ukuran n.
   - Selanjutnya, program meminta input nilai kartu dari pengguna.

- Setiap nilai kartu yang dimasukkan akan dicek validitasnya. Jika nilai kartu adalah "J", "Q", "K", atau "A", maka nilai kartu akan diubah menjadi 11, 12, 13, atau 1 sesuai dengan aturan permainan kartu. Jika nilai kartu adalah angka, maka akan dilakukan konversi ke integer menggunakan fungsi atoi(). Jika nilai kartu tidak valid, program akan mengeluarkan pesan "Invalid input" dan menghentikan eksekusi.
- Setelah semua nilai kartu dimasukkan, program akan memanggil fungsi countSwaps untuk menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu.
- Terakhir, program akan menampilkan output berupa jumlah minimal langkah pertukaran yang diperlukan.
- Sebelum program berakhir, memori yang dialokasikan untuk array cards akan didealokasi menggunakan fungsi free().

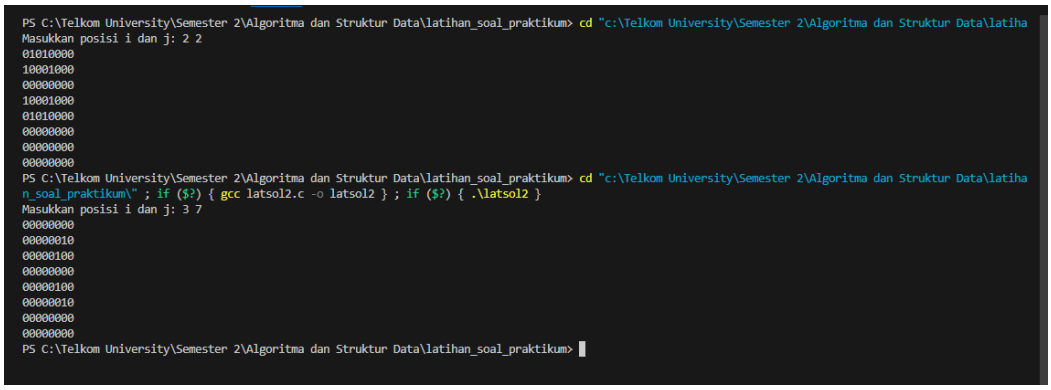
No 2

Source Code

```
#include <stdio.h>

int isValidPosition(int x, int y, int size) {
    return (x >= 0 && x < size && y >= 0 && y < size);
}

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    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 ni = i + dx[k];
        int nj = j + dy[k];

        if (isValidPosition(ni, nj, size)) {
            chessBoard[ni * size + nj] = 1;
        }
    }
}

int main() {
    int i, j;
    printf("Masukkan posisi i dan j: ");
```

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

    int size = 8;
    int chessBoard[size][size];
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            chessBoard[x][y] = 0;
        }
    }

    koboImaginaryChess(i, j, size, (int *)chessBoard);

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

    return 0;
}
```

SS Output



Penjelasan

```
// Fungsi untuk menentukan apakah posisi (x, y) berada dalam papan catur berukuran size x size
int isValidPosition(int x, int y, int size) {
    return (x >= 0 && x < size && y >= 0 && y < size);
}

// Fungsi untuk mengisi array chessBoard dengan 1 pada posisi yang dapat dicapai oleh kuda
void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Mengatur langkah-langkah yang mungkin untuk kuda
    int dx[] = {-2, -1, 1, 2, 2, 1, -1, -2};
    int dy[] = {1, 2, 2, 1, -1, -2, -2, -1};

    // Memeriksa setiap langkah yang mungkin
    for (int k = 0; k < 8; k++) {
        int ni = i + dx[k];
        int nj = j + dy[k];

        // Memeriksa apakah posisi baru valid
        if (isValidPosition(ni, nj, size)) {
```

```
            // Mengisi posisi baru dengan 1 pada array chessBoard
            chessBoard[ni * size + nj] = 1;
        }
    }
}

int main() {
    int i, j;
    // Membaca input i dan j
    printf("Masukkan posisi i dan j: ");
    scanf("%d %d", &i, &j);

    // Inisialisasi array chessBoard dengan nilai 0
    int size = 8;
    int chessBoard[size][size];
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            chessBoard[x][y] = 0;
        }
    }

    // Memanggil fungsi koboImaginaryChess untuk mengisi array chessBoard
    koboImaginaryChess(i, j, size, (int *)chessBoard);

    // Menampilkan array chessBoard
```

```
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d", chessBoard[x][y]);
        }
        printf("\n");
    }

    return 0;
}
```

Bonus

Source Code

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

// Fungsi untuk menampilkan urutan kartu
void printCards(int *cards, int n) {
    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 if (cards[i] == 1) {
            printf("A ");
        } else {
            printf("%d ", cards[i]);
        }
    }
    printf("\n");
}

// Fungsi untuk menghitung jumlah langkah pertukaran yang diperlukan
```

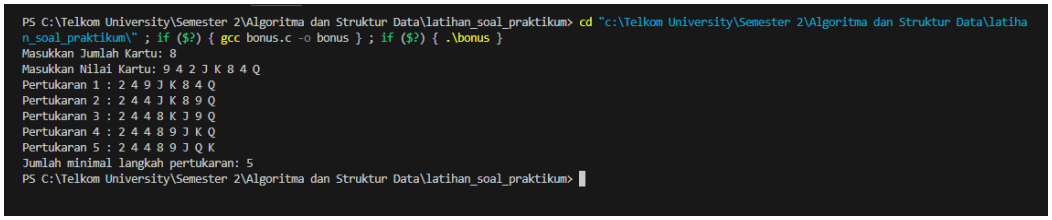
```
int countSwaps(int *cards, int n) {
    int swaps = 0;

    // Menggunakan algoritma Selection Sort untuk mengurutkan kartu
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (cards[j] < cards[minIndex]) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            // Jika kartu dengan nilai terkecil tidak berada di posisi saat ini, lakukan pertukaran
            int temp = cards[i];
            cards[i] = cards[minIndex];
            cards[minIndex] = temp;
            swaps++;
            printf("Pertukaran %d : ", swaps);
            printCards(cards, n);
        }
    }

    return swaps;
}
```

```
int main() {
    int n;
    // Meminta input jumlah kartu
    printf("Masukkan Jumlah Kartu: ");
    if (scanf("%d", &n) != 1) {
        printf("Invalid input\n");
        return 1;
    }

    int *cards = (int *)malloc(n * sizeof(int));
    if (cards == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Membaca nilai kartu
    printf("Masukkan Nilai Kartu: ");
    for (int i = 0; i < n; i++) {
        char value[10];
        if (scanf("%s", value) != 1) {
            printf("Invalid input\n");
            free(cards);
            return 1;
        }
        if (strcmp(value, "J") == 0) {
```

```
        cards[i] = 11;
    } else if (strcmp(value, "Q") == 0) {
        cards[i] = 12;
    } else if (strcmp(value, "K") == 0) {
        cards[i] = 13;
    } else if (strcmp(value, "A") == 0) {
        cards[i] = 1;
    } else {
        cards[i] = atoi(value);
        if (cards[i] < 1 || cards[i] > 10) {
            printf("Invalid input\n");
            free(cards);
            return 1;
        }
    }
}

// Menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu
int minSwaps = countSwaps(cards, n);

// Menampilkan output sesuai yang diharapkan
printf("Jumlah minimal langkah pertukaran: %d\n", minSwaps);

free(cards); // Dealokasi memori
return 0;
```

}

SS Output



Penjelasan

1. Fungsi printCards digunakan untuk mencetak urutan kartu. Pada fungsi ini, setiap nilai kartu akan diperiksa, dan jika nilainya adalah 11, 12, 13, atau 1, maka akan dicetak sebagai "J", "Q", "K", atau "A" secara berurutan. Jika bukan, maka nilai kartu akan dicetak sesuai dengan nilainya.

2. Fungsi countSwaps digunakan untuk menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu. Fungsi ini menggunakan algoritma Selection Sort untuk mengurutkan kartu, dan setiap pertukaran akan dicetak dengan menggunakan fungsi printCards.

3. Pada bagian main, setelah meminta input jumlah kartu, program akan mengalokasikan memori untuk menyimpan nilai kartu. Kemudian, program akan membaca nilai kartu yang dimasukkan oleh pengguna. Jika nilai kartu adalah "J", "Q", "K", atau "A", maka nilai tersebut akan diubah menjadi 11, 12, 13, atau 1. Selain itu, nilai kartu akan diubah menjadi bilangan bulat menggunakan fungsi atoi.

4. Setelah itu, program akan memanggil fungsi countSwaps untuk menghitung jumlah langkah pertukaran yang diperlukan untuk mengurutkan kartu. Hasilnya akan dicetak sebagai jumlah minimal langkah pertukaran yang diperlukan.