NAMA : FELIX SIMAMORA

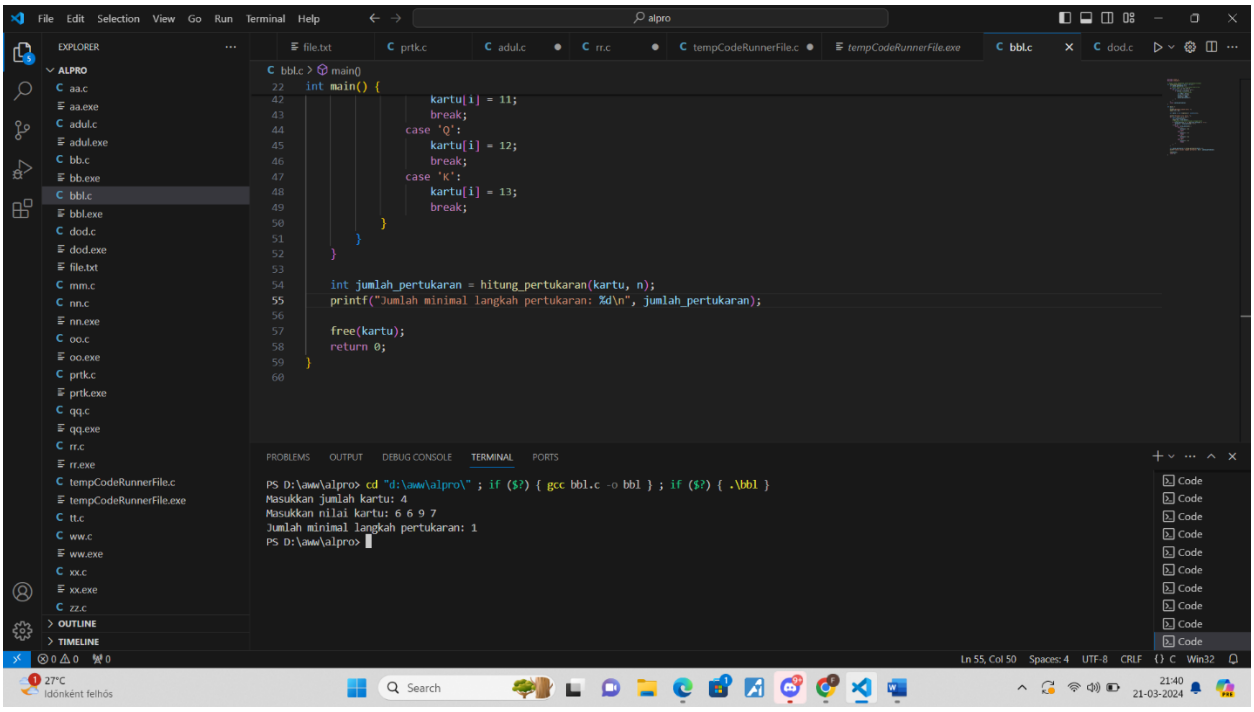
NIM : 1203230068

MATKUL: PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

NO.1

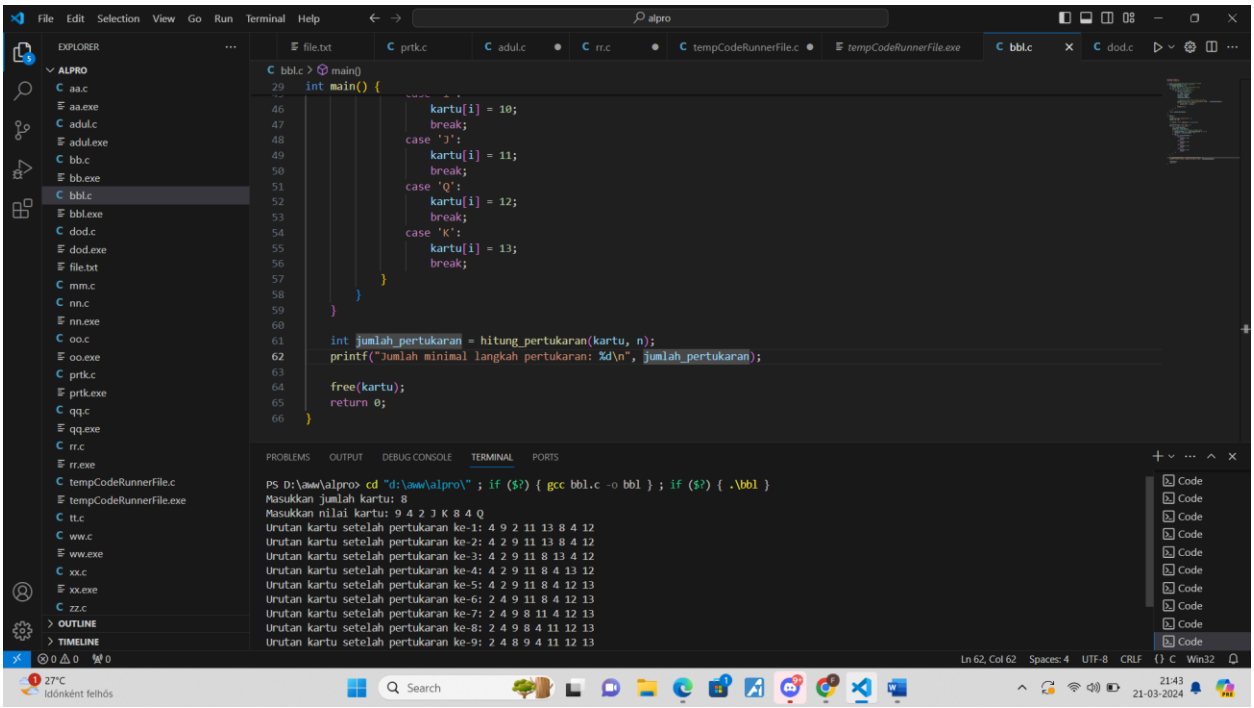
```c
#include <stdio.h>
#include <stdlib.h>

// Fungsi untuk menghitung jumlah pertukaran minimum
int hitung_pertukaran(int arr[], int n) {
    int jumlah_pertukaran = 0;
    // Lakukan bubble sort untuk mengurutkan array
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar posisi elemen
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                jumlah_pertukaran++;
            }
        }
    }
    return jumlah_pertukaran;
}

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

    int *kartu = (int *)malloc(n * sizeof(int));

    printf("Masukkan nilai kartu: ");
    for (int i = 0; i < n; i++) {
        char nilai_kartu[3];
        scanf("%s", nilai_kartu);
        // Mengubah nilai kartu menjadi bilangan bulat
        if (nilai_kartu[0] >= '2' && nilai_kartu[0] <= '9') {
            kartu[i] = nilai_kartu[0] - '0';
        } else {
            switch (nilai_kartu[0]) {
```

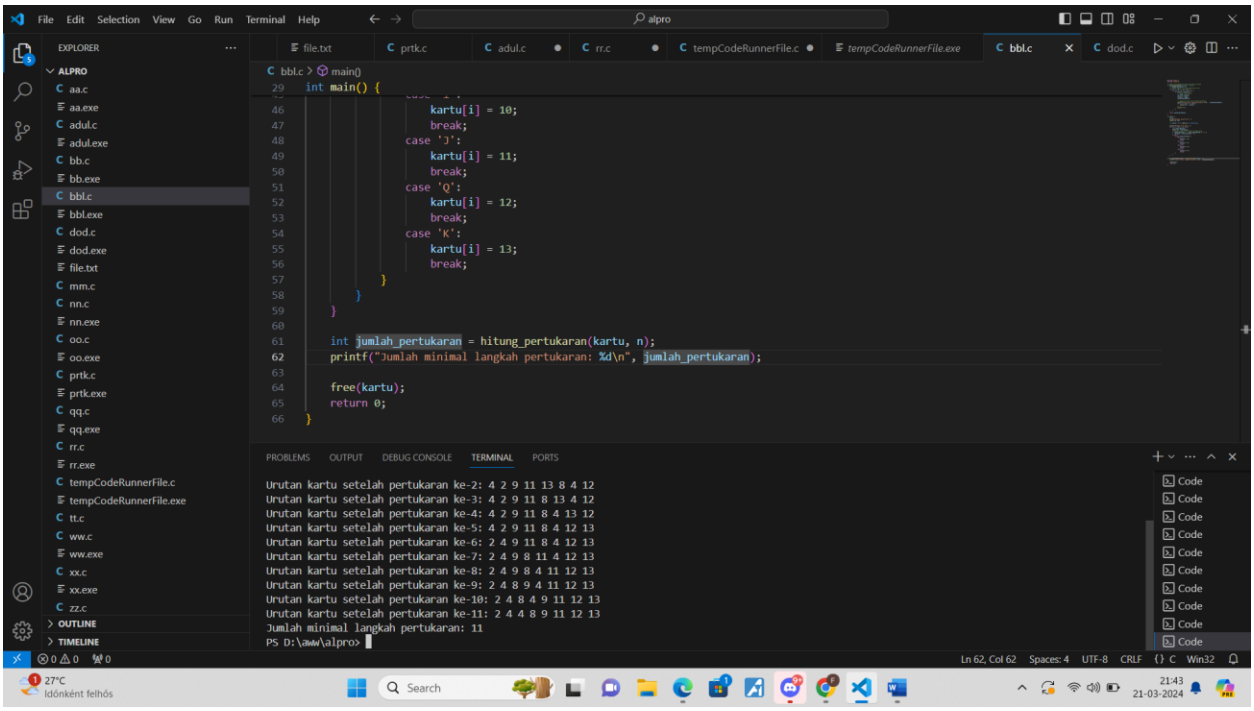
```
                case '1':
                    kartu[i] = 10;
                    break;
                case 'J':
                    kartu[i] = 11;
                    break;
                case 'Q':
                    kartu[i] = 12;
                    break;
                case 'K':
                    kartu[i] = 13;
                    break;
            }
        }
    }

    int jumlah_pertukaran = hitung_pertukaran(kartu, n);
    printf("Jumlah minimal langkah pertukaran: %d\n", jumlah_pertukaran);

    free(kartu);
    return 0;
}
```



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

// Fungsi untuk menghitung jumlah pertukaran minimum
int hitung_pertukaran(int arr[], int n) {
    int jumlah_pertukaran = 0;
    // Lakukan bubble sort untuk mengurutkan array
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar posisi elemen
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                jumlah_pertukaran++;

                // Tampilkan urutan kartu setelah pertukaran
                printf("Urutan kartu setelah pertukaran ke-%d: ",
jumlah_pertukaran);
                for (int k = 0; k < n; k++) {
                    printf("%d ", arr[k]);
                }
                printf("\n");
            }
        }
    }
    return jumlah_pertukaran;
}

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

    int *kartu = (int *)malloc(n * sizeof(int));

    printf("Masukkan nilai kartu: ");
    for (int i = 0; i < n; i++) {
        char nilai_kartu[3];
        scanf("%s", nilai_kartu);
        // Mengubah nilai kartu menjadi bilangan bulat
        if (nilai_kartu[0] >= '2' && nilai_kartu[0] <= '9') {
            kartu[i] = nilai_kartu[0] - '0';
        } else {
            switch (nilai_kartu[0]) {
```

```c
                case '1':
                    kartu[i] = 10;
                    break;
                case 'J':
                    kartu[i] = 11;
                    break;
                case 'Q':
                    kartu[i] = 12;
                    break;
                case 'K':
                    kartu[i] = 13;
                    break;
            }
        }
    }

    int jumlah_pertukaran = hitung_pertukaran(kartu, n);
    printf("Jumlah minimal langkah pertukaran: %d\n", jumlah_pertukaran);

    free(kartu);
    return 0;
}
```

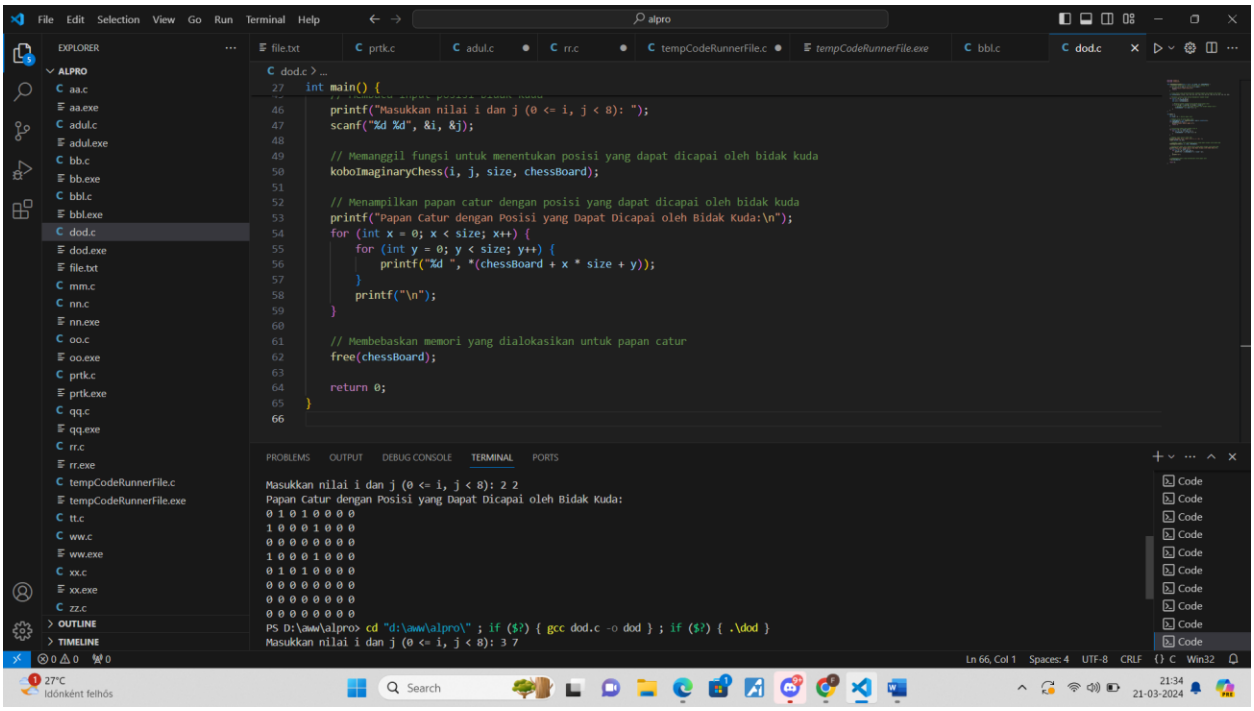


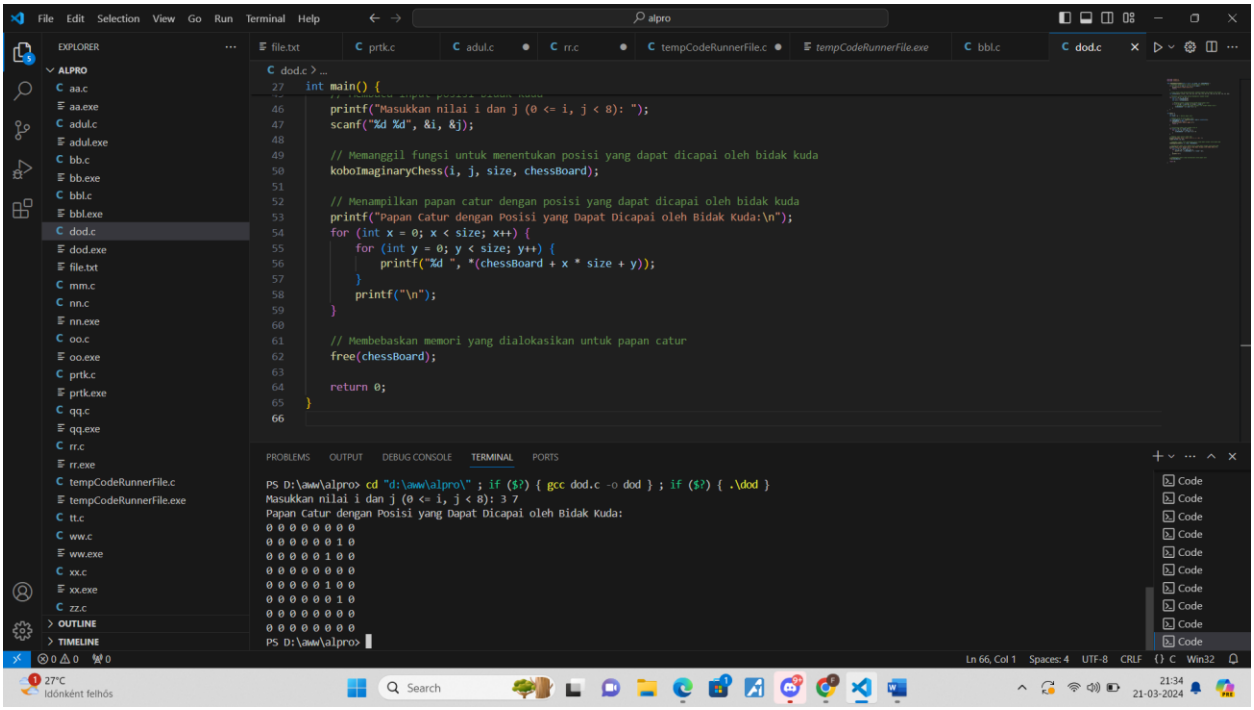


NO.2

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

void koboImaginaryChess(int i, int j, int size, int *chessBoard) {
    // Mengecek apakah posisi i dan j berada dalam rentang valid
    if (i < 0 || i >= size || j < 0 || j >= size) {
        printf("Posisi tidak valid.\n");
        return;
    }

    // Array offset yang merepresentasikan langkah-langkah yang mungkin oleh knight
    int offset[8][2] = {{-2, -1}, {-2, 1}, {-1, -2}, {-1, 2}, {1, -2}, {1, 2}, {2, -1}, {2, 1}};

    // Melakukan iterasi untuk setiap kemungkinan langkah knight
    for (int k = 0; k < 8; k++) {
        int x = i + offset[k][0];
        int y = j + offset[k][1];

        // Mengecek apakah langkah masih berada dalam papan catur
        if (x >= 0 && x < size && y >= 0 && y < size) {
            // Menandai posisi yang dapat dicapai oleh knight dengan nilai 1
```

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

int main() {
    int i, j;
    int size = 8; // Ukuran papan catur

    // Membuat array 2D untuk papan catur
    int *chessBoard = (int *)malloc(size * size * sizeof(int));
    if (chessBoard == NULL) {
        printf("Alokasi memori gagal.\n");
        return 1;
    }

    // Inisialisasi papan catur dengan nilai 0
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            *(chessBoard + x * size + y) = 0;
        }
    }

    // Membaca input posisi bidak kuda
    printf("Masukkan nilai i dan j (0 <= i, j < 8): ");
    scanf("%d %d", &i, &j);

    // Memanggil fungsi untuk menentukan posisi yang dapat dicapai oleh bidak
kuda
    koboImaginaryChess(i, j, size, chessBoard);

    // Menampilkan papan catur dengan posisi yang dapat dicapai oleh bidak kuda
    printf("Papan Catur dengan Posisi yang Dapat Dicapai oleh Bidak Kuda:\n");
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            printf("%d ", *(chessBoard + x * size + y));
        }
        printf("\n");
    }

    // Membebaskan memori yang dialokasikan untuk papan catur
    free(chessBoard);

    return 0;
}
```





KOMPONEN PENILAIAN	YA	TIDAK
Soal 1 sesuai dengan output yang diinginkan	YA	
Soal 2 sesuai dengan output yang diinginkan	YA	
Bonus soal 1 dikerjakan	YA	