Tugas Kelompok Kedua

Group 2
MUDIANTO
2702358821

JASON ALEXANDER CHRISTOPHORUS
2702360656

TIFFANY WIDJAJA
2702349804

CORNELIA NATHALIE WAU
2702360555

BAYU FITO SANJAYA
2702359364

Files included with this paper

no1.c	no1.exe
no2.c	no2.exe

Soal 1

```c
#include <stdio.h>
#define GREEN "\033[0;32m"
#define WHITEBOLD "\033[1;37m"
#define RESET "\033[0;0m"

//color code format "\033[(num);(color)m". num = 0 untuk teks biasa, 1 untuk bold.

//mengganti \n di akhir string yang didapat dari fgets menjadi \0
void nullify(char s[], int bufsize) {
    for(int i = 0; i < bufsize; i++) {
        if(s[i] == '\n') {
            s[i] = '\0';
            break;
        }
    }
}

//memprint message untuk meminta input pengguna
void displayMessage() {
    printf("Masukkan total pembelian Anda (masukkan angka bulat saja): Rp");
}

//memprint garis
void line() {
    printf("-------------------------------------------------------------\n");
}

void prosesTransaksi(int totalBeli, float * arrayTransaksiAkhir);
int strToInt(char s[]);

int main() {
    char totalPembelianStr[255]; // buffer
    int totalPembelian;
    float hasilTransaksiAkhir[4];

    displayMessage();
    while(fgets(totalPembelianStr, 255, stdin)) {
        line();
        nullify(totalPembelianStr, 255);
        totalPembelian = strToInt(totalPembelianStr);

        //cek input kosong
        if(totalPembelianStr[0] == '\0') {
            printf("Maaf, input Anda tidak boleh kosong.\n\n");
            displayMessage();
            continue;
        }

        //cek input invalid
        if(strToInt(totalPembelianStr) == -1) {
            printf("Maaf, input Anda mengandung karakter invalid. Karakter yang
diperbolehkan hanya angka (0-9).\n\n");
            displayMessage();
            continue;
        }
        //printf("total: %d", strToInt(totalPembelian));
        prosesTransaksi(totalPembelian, hasilTransaksiAkhir);
        float awal = hasilTransaksiAkhir[0];
        float kupon = hasilTransaksiAkhir[1];
        float diskon = hasilTransaksiAkhir[2];
        float akhir = hasilTransaksiAkhir[3];
```

```
        printf("%sTRANSAKSI\n%s", WHITEBOLD, RESET);
        line();
        printf("Harga awal                  : Rp%.0f\n", awal);
        printf("Kupon undian yang didapat   : %.0f\n", kupon);
        if(kupon > 5) {
            printf("%sDiskon yang didapat         : 25.00%% (max. 5 kupon)\n%s", GREEN,
RESET);
        } else if(kupon > 0) {
            printf("%sDiskon yang didapat         : Rp%.2f%%\n%s", GREEN, diskon * 100,
RESET);
        } else {
            printf("Diskon yang didapat         : Rp0.00%%\n");
        }
        printf("Anda menghemat              : Rp%.0f\n", awal-akhir);
        printf("Harga akhir                 : Rp%.0f\n", akhir);
        line();
        printf("\n");
        displayMessage();
    }
    return 0;
}

//mengubah string menjadi integer (karena menggunakan fgets)
int strToInt(char s[]) {
    //returnVal akan mengakumulasi nilai ASCII dari angka yang dimasukkan - 48 untuk
mendapatkan nilai aslinya, misal '1' == ASCII 49 - 48 = 1.
    int returnVal = 0;
    for(int i = 0; s[i] != '\0'; i++) {
        if(s[i] >= 48 && s[i] <= 57) {
            //angka yang sudah ada di"shift" ke kiri dengan mengalikan 10 untuk menyediakan
ruang untuk digit selanjutnya.
            returnVal = (returnVal * 10) + (s[i] - 48);
        } else {
            return -1; // error code -1;
        }
    }
    return returnVal;
}

//pass by reference int array (memasukkan nilai ke array yg dipass ke function)
void prosesTransaksi(int totalBeli, float * arrayTransaksiAkhir) {
    float hargaAwal = totalBeli;
    float jmlKupon; // 1 per Rp100.000,00
    float diskon; // 5% per kupon, dimaksimalkan 25% atau 5 kupon supaya realistis
    float hargaAkhir;

    //menentukan jumlah kupon
    jmlKupon = totalBeli / 100000; // tidak diapa-apakan karena sudah mengambil hasil
bulatnya saja (seperti Math.floor di JS)

    //menentukan diskon dr kupon (0.05 -> 5%)
    diskon = (float)jmlKupon * 0.05;
    if(diskon > 0.25) {
        diskon = 0.25;
    }

    //harga akhir
    hargaAkhir = (1-diskon)*hargaAwal;

    //memasukkan nilai akhir ke array yang dipass ke function.
    arrayTransaksiAkhir[0] = hargaAwal;
    arrayTransaksiAkhir[1] = jmlKupon;
    arrayTransaksiAkhir[2] = diskon;
    arrayTransaksiAkhir[3] = hargaAkhir;
}
```

Soal 2

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

// Define a constant for the maximum text length
#define TEXT_LENGTH 50

// Define a constant for the maximum number of employees
#define MAX_EMPLOYEES 100

// Function to get a valid integer input
int get_valid_int_input(const char* prompt)
{
    int input;
    while (1)
    {
        printf("%s", prompt);
        if ((scanf("%d", &input) == 1) && input > 0)
        {
            // Valid input, break out of the loop
            while (getchar() != '\n'); // Clear the input buffer
            return input;
        }
        else
        {
            // Invalid input, clear the input buffer
            while (getchar() != '\n');
            printf("Input invalid. Mohon masukkan integer yang valid.\n");
        }
    }
}

// Function to get a valid string input with buffer size checking
void get_valid_string_input(const char* prompt, char* buffer, int bufferSize)
{
    while (1)
    {
        printf("%s", prompt);
        if (scanf("%s", buffer) == 1)
        {
            // Valid input, check if input length exceeds buffer size
            if (strlen(buffer) > bufferSize - 1)
            {
                printf("Input melebihi jumlah karakter maksimum %d. Mohon
masukkan string yang lebih pendek.\n", bufferSize - 1);
                while (getchar() != '\n'); // Clear the input buffer
            }
            else
```

```
            {
                // Input is within the buffer size limit, break out of the
loop
                while (getchar() != '\n'); // Clear the input buffer
                return;
            }
        }
        else
        {
            // Invalid input, clear the input buffer
            while (getchar() != '\n');
            printf("Input invalid. Mohon masukkan input yang valid.\n");
        }
    }
}

// Function to get a valid group (D1/D2/D3) input
void get_valid_group_input(const char* prompt, char* buffer)
{
    while (1)
    {
        printf("%s", prompt);
        if ((scanf("%s", buffer) == 1) &&
           ((strcmp(buffer, "D1") == 0) || (strcmp(buffer, "D2") == 0) ||
(strcmp(buffer, "D3") == 0)) )
        {
```

```c
// Valid input, break out of the loop
            while (getchar() != '\n'); // Clear the input buffer
            return;
        }
        else
        {
            // Invalid input, clear the input buffer
            while (getchar() != '\n');
            printf("Input invalid. Mohon masukkan D1, D2, atau D3 saja.\n");
        }
    }
}

// Define a struct for Employee
struct Employee
{
    int ID;
    char name[TEXT_LENGTH];
    char address[TEXT_LENGTH];
    int phone_number;
    char job_title[TEXT_LENGTH];
    char group[3]; // 3 Options: D1, D2, or D3 (2 characters + null terminator)
    long int base_salary;
    short int overtime_hours;
    long int total_monthly_salary;
};

int main()
{
    int num_employees;

    // Ask the user for the number of employees to add
    num_employees = get_valid_int_input("Masukkan jumlah pegawai yang akan didata: ");

    // Check if the number of employees exceeds the maximum
    if (num_employees > MAX_EMPLOYEES)
    {
        printf("Error: Jumlah pegawai melebih maksimum %d orang.\n", MAX_EMPLOYEES);
        return 1;
    }

    // Declare an array to store employee data
    struct Employee employee_list[num_employees];

    // Initialize each element of the array to zero
    for (int i = 0; i < num_employees; i++)
    {
        memset(&employee_list[i], 0, sizeof(struct Employee));
    }

    // Add code to ask for input of each employee's details (Part 1)
    printf("\033[31mInput 1\033[0m\n");
    for (int i = 0; i < num_employees; i++)
    {
        printf("\033[32mPegawai no.%d:\033[0m\n", i + 1);

        // Get valid integer input for ID
        employee_list[i].ID = get_valid_int_input("NIP: ");

        // Get valid string input for Name, Address, Job Title, and Group
        get_valid_string_input("Nama: ", employee_list[i].name, TEXT_LENGTH);
        get_valid_string_input("Alamat: ", employee_list[i].address, TEXT_LENGTH);

        // Get valid integer input for phone number
        employee_list[i].phone_number = get_valid_int_input("No HP: ");

        get_valid_string_input("Jabatan: ", employee_list[i].job_title, TEXT_LENGTH);
        get_valid_group_input("Golongan (D1/D2/D3): ", employee_list[i].group);

        // Determine base salary
```

```
if (strcmp(employee_list[i].group, "D1") == 0)
        {
            employee_list[i].base_salary = employee_list[i].total_monthly_salary = 2500000;
        }
        else if (strcmp(employee_list[i].group, "D2") == 0)
        {
            employee_list[i].base_salary = employee_list[i].total_monthly_salary = 2000000;
        }
        else if (strcmp(employee_list[i].group, "D3") == 0)
        {
            employee_list[i].base_salary = employee_list[i].total_monthly_salary = 1500000;
        }
    }

    // Add code to search for an employee by ID and group (Part 2)
    printf("\033[31mInput 2\033[0m\n");
    char exit_part_2[100] = "Y"; // Y = continue, case sensitive.
    while (strcmp(exit_part_2, "Y") == 0)
    {
        // Prompt the user for the target ID to search for
        int target_ID = 0;
        char target_group[3] = "";
        target_ID = get_valid_int_input("NIP: ");
        get_valid_group_input("Golongan (D1/D2/D3): ", target_group);

        // Search for the employee by ID
        char found = 0; // Flag to indicate if the employee was found
        char mismatch = 0; // Flag to indicate if there are any identity mismatch errors
        for (int i = 0; i < num_employees; i++)
        {
            if (employee_list[i].ID == target_ID)
            {
                // Employee found, match the descriptions
                if (strcmp(employee_list[i].group, target_group) != 0)
                {
                    mismatch = 1;
                    printf("\033[33mIdentitas tidak cocok, NIP pegawai ditemukan tapi dalam
golongan yang berbeda.\033[0m\n");
                    break;
                }

                // Overtime hours
                printf("Jam lembur: ");
                scanf("%d", &employee_list[i].overtime_hours);

                // Calculate salary
                if (strcmp(employee_list[i].group, "D1") == 0)
                {
                    employee_list[i].total_monthly_salary = employee_list[i].base_salary +
employee_list[i].overtime_hours * 10000;
                }
                else if (strcmp(employee_list[i].group, "D2") == 0)
                {
                    employee_list[i].total_monthly_salary = employee_list[i].base_salary +
employee_list[i].overtime_hours * 5000;
                }
                else if (strcmp(employee_list[i].group, "D3") == 0)
                {
                    employee_list[i].total_monthly_salary = employee_list[i].base_salary +
employee_list[i].overtime_hours * 2500;
                }
                found = 1; // Set the flag to indicate that the employee was found
                break; // Exit the loop once the employee is found
            }
        }
        // Display a message if the employee was not found
        if (!found && (mismatch == 0))
        {
            printf("\033[33mPegawai dengan NIP %d tidak ditemukan.\033[0m\n", target_ID);
        }
        get_valid_string_input("Ada pegawai lain yang lembur? (Y = yes, case-sensitive)",
exit_part_2, 100);
    }

    // Print the list of employees and their details (Part 3, Output)
    printf("\033[31mDaftar Pegawai:\033[0m\n");
    for (int j = 0; j < num_employees; j++)
    {
        printf("\033[32mPegawai #%d:\033[0m\n", j + 1);
        printf("NIP: %d\n", employee_list[j].ID);
        printf("Nama: %s\n", employee_list[j].name);
        printf("Alamat: %s\n", employee_list[j].address);
        printf("No HP: %d\n", employee_list[j].phone_number);
        printf("Jabatan: %s\n", employee_list[j].job_title);
        printf("Golongan: %s\n", employee_list[j].group);
        printf("Gaji pokok: Rp%ld\n", employee_list[j].base_salary); // Use %ld for long int
        printf("Jam lembur: %d\n", employee_list[j].overtime_hours);
        printf("Total gaji bulan ini: Rp%ld\n", employee_list[j].total_monthly_salary); //
Use %ld for long int
    }
    getchar();
    return 0;
}
```