TUGAS OTH

Nama: danendra urdha bhanu cetta harenndy

Nim : 1203230110

Untuk soal No. 1

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

// Struktur Node yang disediakan
typedef struct Node {
  char data;
  struct Node* next;
} Node;

int main() {
  Node n =  {'I', NULL};
  Node* f = (Node*)malloc(sizeof(Node));
  f->data = 'N';
  Node* o = (Node*)malloc(sizeof(Node));
  o->data = 'F';
  Node* r = (Node*)malloc(sizeof(Node));
  r->data = '0';
  Node* m = (Node*)malloc(sizeof(Node));
  m->data = 'R';
  Node* a = (Node*)malloc(sizeof(Node));
  a->data = 'M';
  Node* t = (Node*)malloc(sizeof(Node));
  t->data = 'A';
  Node* i = (Node*)malloc(sizeof(Node));
  i->data = 'T';
  Node* k = (Node*)malloc(sizeof(Node));
  k->data = 'I';
  Node* a2 = (Node*)malloc(sizeof(Node));
  a2->data = 'K';
  Node* i3 =  (Node*)malloc(sizeof(Node));
  i3->data = 'A';

  n.next = f;
  f->next = o;
  o->next = r;
  r->next = m;
  m->next = a;
  a->next = t;
  t->next = i;
  i->next = k;
  k->next = a2;
```

```
  a2->next = i3;
  i3->next = NULL;

  Node* l3 = &n;

  char word[13];
  int index = 0;
  while (l3 != NULL) {
    word[index++] = l3->data;
    l3 = l3->next;
  }
  word[index] = '\0';

  printf("%s\n", word);
  free(f);
  free(o);
  free(r);
  free(m);
  free(a);
  free(t);
  free(i);
  free(k);
  free(a2);

  return 0;
}
```

Output:

```
PS C:\Users\danen\OneDrive\Desktop\contoh2 variable c> cd "c:\Users\danen\
hayata.c -o logakbahayata } ; if ($?) { .\logakbahayata }
INF0RMATIKA
PS C:\Users\danen\OneDrive\Desktop\contoh2 variable c>
```

Penjelasan :

Pada program ini saya menggunakan 2 include, yaitu stdio.h dan stdlib.h

Untuk struknya saya menggunakan typedef struct untuk mempersinkat dan memperjelas jalanya pemrograman dan juga untuk menyimpan data pada setiap simpul

Pada bagian awal fungsi main saya mengunnakan malloc untuk mengalokasikan memory untuk sebuah objek yang bernilai Node

Kemudian saya menambahkan pada sitiap charakter -> next untuk mengatur jalan pada pola agar membentuk output yang diinginkan

Pada baris berikutnya saya mengunakan Node* l3 = &n; untuk mendeklarasikan pointer l3 yang menunjuk ke alamat memori dari variabel n bertipe Node

Pada bris di bawahnya saya menggunakan sebuah perulangan untuk untuk mengisi array word dengan karakter yang disimpan di setiap simpul dari linked list yang dimulai dari simpul yang ditunjuk oleh pointer l3

Kemudian saya tambakan printf untuk mengoutput data yang telah terbentuk

Dan saya akhiri dengan return 0;

No2.

```
#include <stdio.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int sum_stack_a[n + 1];
    int sum_stack_b[m + 1];

    sum_stack_a[0] = 0;
    for (int i = 1; i <= n; i++) {
        sum_stack_a[i] = sum_stack_a[i - 1] + a[i - 1];
    }

    sum_stack_b[0] = 0;
    for (int i = 1; i <= m; i++) {
        sum_stack_b[i] = sum_stack_b[i - 1] + b[i - 1];
    }

   int max_count = 0;
    int j = m;

    for (int i = 0; i <= n; i++) {
        if (sum_stack_a[i] > maxSum) {
            break;
        }

        while (j > 0 && sum_stack_a[i] + sum_stack_b[j] > maxSum) {
            j--;
        }

        max_count = max(max_count, i + j);
    }

    return max_count;
}

int main() {
    int g;
```

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

    for (int t = 0; t < g; t++) {
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);

        int a[n];
        int b[m];

        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]);
        }

        for (int i = 0; i < m; i++) {
            scanf("%d", &b[i]);
        }

        printf("%d\n", twoStacks(maxSum, a, n, b, m));
    }

    return 0;
}
```

Output:

```
1
5 4 10
4 2 4 6 1
2 1 8 5
4
PS C:\Users\danen\On
wakmucak1.c -o sanga
1
5 4 11
4 5 2 1 1
3 1 1 2
5
```

Penjelasan :

Pada program ini saya hanya mengggunakan include <stdio.h>

Kemudian saya mamebuat fungsi max yang berisi integer a dan b dan pada fungsi itu juga saya menambahkan ekpresi ternary yang merupakan cara singkat untuk menulis kondisional if-else

Kemudian saya menbuat fungsi twoStack yang berisi inr maxsum,int a[],int n, int b[]dan int m

Fungsi tersebut berfungsi untuk menemukan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan tanpa melampaui maxSum

Di dalam fungsi ini saya menambahkan dua array tambahan, sum_stack_a dan sum_stack_b, yang masing-masing menyimpan jumlah kumulatif dari elemen-elemen tumpukan pertama dan kedua.

Berikut adalah hal hal yang saya lakukan di dalam fungsi tersebut:

-Melakukan iterasi melalui tumpukan pertama dan mencatat jumlah kumulatif dari elemen-elemen tumpukan tersebut dalam sum_stack_a.

-Melakukan hal yang sama untuk tumpukan kedua dan menyimpan jumlah kumulatifnya dalam sum_stack_b.

-Memulai iterasi dari awal tumpukan pertama dan menemukan jumlah elemen maksimum yang dapat diambil dari kedua tumpukan dengan mempertimbangkan batas maxSum.

-Iterasi ini menggunakan dua pointer, satu untuk tumpukan pertama (i) dan satu untuk tumpukan kedua (j).

-Selama iterasi, kita mempertahankan pointer untuk tumpukan kedua (j) sehingga total elemen dari kedua tumpukan tidak melampaui maxSum.

-Setiap kali kita menemukan jumlah elemen yang memenuhi kriteria, kita bandingkan dengan nilai maksimum yang tercatat sebelumnya (max_count) dan memperbarui jika diperlukan.

-Mengembalikan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan.

Dan yang terakhir adalah int main / fungsi utama

Di sini saya mendeklarasikan variable g yang akan digunakan untuk menytyimpan jumlah kasus uji.

Di fungsi ini juga saya mengunakan loop yang bertujuan untuk Memanggil fungsi twoStacks dengan parameter maxSum, a, n, b, dan m untuk menyelesaikan kasus uji tersebut dan mencetak hasilnya ke output

Dan saya akhiri dengan return studio.h

Visualisasi alur penyelesaian jika input :

1

5 4 11

4 5 2 1 1

3 1 1 2

Berikut adalah bentuk visualnya:

Tumpukan Pertama (a):

[4]

[5]

[2]

[1]

[1]

Tumpukan Kedua (b):

[3]

[1]

[1]

[2]