NAMA : FARREL AHNAF KHAYLA PRAPTAMA

NIM :1203230100

KLS : IF 03-03

TUGAS OTH

Source code 1

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

struct Node {
    char* alphabet;
    struct Node* link;
};

int main() {

    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;


    l1.link = NULL;
    l1.alphabet = "F";

    l2.link = NULL;
    l2.alphabet = "M";

    l3.link = NULL;
    l3.alphabet = "A";

    l4.link = NULL;
    l4.alphabet = "I";

    l5.link = NULL;
    l5.alphabet = "K";

    l6.link = NULL;
    l6.alphabet = "T";

    l7.link = NULL;
    l7.alphabet = "N";

    l8.link = NULL;
    l8.alphabet = "O";

    l9.link = NULL;
    l9.alphabet = "R";
```

```
    l7.link = &l1;
    l1.link = &l8;
    l8.link = &l2;
    l2.link = &l5;
    l5.link = &l3;
    l3.link = &l6;
    l6.link = &l9;
    l9.link = &l4;
    l4.link = &l7;


    l3ptr = &l7;


    printf("%s", l3.link->link->link->alphabet);
    printf("%s", l3.link->link->link->link->alphabet);
    printf("%s", l3.link->link->link->link->link->alphabet);
    printf("%s", l3.link->link->link->link->link->link->alphabet);
    printf("%s", l3.link->link->alphabet);
    printf("%s", l3.link->link->link->link->link->link->link->alphabet);
    printf("%s", l3.alphabet);
    printf("%s", l3.link->alphabet);
    printf("%s", l3.link->link->link->alphabet);
    printf("%s", l3.link->link->link->link->link->link->link->link->alphabet);
    printf("%s", l3.alphabet);


    return 0;
}
```

Penjelasan

#include <stdio.h>

#include <stdlib.h>

Baris pertama dan kedua adalah direktif praprosesor yang memasukkan file header standar stdio.h dan stdlib.h ke dalam program. File stdio.h digunakan untuk fungsi input/output standar, sementara stdlib.h digunakan untuk alokasi memori dinamis.

struct Node {

  char* alphabet;

  struct Node* link;

};

Mendefinisikan struktur Node yang memiliki dua anggota: alphabet yang bertipe pointer ke karakter (string) dan link yang bertipe pointer ke struktur Node itu sendiri. Struktur ini akan digunakan untuk membentuk linked list.

int main() {

Fungsi utama program dimulai.

struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;

struct Node *link, *l3ptr;

Mendeklarasikan beberapa variabel dari tipe Node dan dua pointer ke Node.

l1.link = NULL;

l1.alphabet = "F";

Menginisialisasi l1 dengan nilai link yang menunjuk ke NULL dan nilai alphabet berupa string "F".

// Menginisialisasi variabel lainnya serupa dengan l1

Sama seperti langkah sebelumnya, melakukan inisialisasi untuk variabel l2 hingga l9 dengan nilai yang sesuai.

l7.link = &l1;

l1.link = &l8;

l8.link = &l2;

l2.link = &l5;

l5.link = &l3;

l3.link = &l6;

l6.link = &l9;

l9.link = &l4;

l4.link = &l7;

Membuat linked list dengan menghubungkan setiap node secara berurutan satu sama lain.

l3ptr = &l7;

Mengatur pointer l3ptr untuk menunjuk ke node l7.

printf("%s", l3.link->link->link->alphabet);

Mencetak nilai dari karakter yang ditunjuk oleh pointer pada node ketiga dari node l3 dalam urutan linked list.

// printf() statements berikutnya melakukan hal yang serupa dengan yang dijelaskan pada poin 9, hanya berbeda dalam perhitungan jumlah loncatan node yang dilakukan untuk mencapai node yang diinginkan.

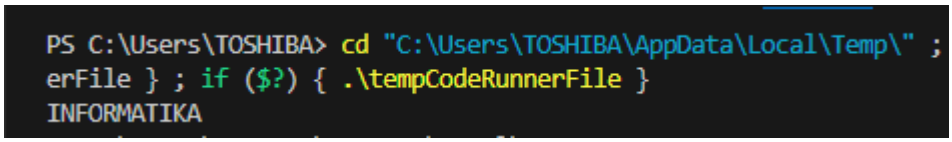
return 0;

Mengembalikan nilai 0 yang menandakan keluar dari program secara normal.

}

Tutup dari fungsi main.

Output 1



Source code 2

```c
#include <stdio.h>

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

int twoStacks(int maxSum, int a[], int m, int b[], int n) {
    int total = 0, count = 0, maxCount = 0;
    int i = 0, j = 0;


    while (i < m && total + a[i] <= maxSum) {
        total += a[i++];
        count++;
        maxCount = max(maxCount, count);
    }


    while (j < n && i >= 0) {
        total += b[j++];
        count++;


        while (total > maxSum && i > 0) {
            i--;
            total -= a[i];
            count--;
        }

        if (total <= maxSum) {
            maxCount = max(maxCount, count);
        }
    }

    return maxCount;
}
```

```
int main() {
    int games;
    scanf("%d", &games);

    for (int g = 0; g < games; g++) {
        int m, n, maxSum;
        scanf("%d %d %d", &m, &n, &maxSum);
        int a[m], b[n];


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

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


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

    return 0;
}
```

Penjelasan

#include <stdio.h>

Direktif praprosesor yang memasukkan file header standar stdio.h, yang diperlukan untuk fungsi input/output standar seperti printf dan scanf.

int max(int a, int b) {

  return a > b ? a : b;

}

Deklarasi fungsi max yang mengembalikan nilai maksimum dari dua bilangan a dan b.

int twoStacks(int maxSum, int a[], int m, int b[], int n) {

Deklarasi fungsi twoStacks yang mengambil parameter maxSum (jumlah maksimum yang diizinkan), array a dan b yang masing-masing merepresentasikan elemen-elemen dari dua stack, serta ukuran masing-masing stack (m untuk stack pertama dan n untuk stack kedua).

```
int total = 0, count = 0, maxCount = 0;
int i = 0, j = 0;
```

Deklarasi beberapa variabel yang akan digunakan dalam proses penghitungan jumlah maksimum elemen yang dapat diambil.

```
while (i < m && total + a[i] <= maxSum) {
    total += a[i++];
    count++;
    maxCount = max(maxCount, count);
}
```

Perulangan while pertama untuk mengambil sebanyak mungkin elemen dari stack pertama (a) hingga total melebihi atau sama dengan maxSum.

```
while (j < n && i >= 0) {
```

Perulangan while kedua untuk mempertimbangkan elemen dari stack kedua (b) sambil memperhitungkan kembali elemen dari stack pertama jika total melebihi maxSum.

```
while (total > maxSum && i > 0) {
    i--;
    total -= a[i];
    count--;
```

```
}
```

Perulangan dalam perulangan yang mengurangi elemen dari stack pertama jika total melebihi maxSum.

```
if (total <= maxSum) {
   maxCount = max(maxCount, count);
}
```

perulangan yang mengurangi elemen dari stack pertama jika total melebihi maxSum.

```
if (total <= maxSum) {
   maxCount = max(maxCount, count);
}
```

Memperbarui nilai maxCount jika total kurang dari atau sama dengan maxSum.

```
return maxCount;
```

Mengembalikan nilai maxCount, yang merupakan jumlah maksimum elemen yang dapat diambil dari kedua stack.

```
int main() {
   ...
}
```

Fungsi utama program dimulai.

```
int games;
```

scanf("%d", &games);

Meminta input jumlah permainan dari pengguna.

for (int g = 0; g < games; g++) {

...

}

Perulangan for untuk setiap permainan.

int m, n, maxSum;

scanf("%d %d %d", &m, &n, &maxSum);

Meminta input ukuran stack dan maxSum untuk permainan saat ini.

int a[m], b[n];

Mendeklarasikan array a dan b dengan ukuran sesuai input.

for (int i = 0; i < m; i++) {

scanf("%d", &a[i]);

}

Meminta input elemen untuk stack pertama (a).

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

scanf("%d", &b[i]);

}

Meminta input elemen untuk stack kedua (b).

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

Memanggil fungsi twoStacks untuk menghitung dan mencetak jumlah maksimum elemen yang dapat diambil dari kedua stack.

return 0;

Mengembalikan nilai 0 yang menandakan keluar dari program secara normal.

}

Tutup dari fungsi main.

Output 2