Nama: Richie

Nim: 12003230064

Tugas struct dan Stack

1.Asisten Sherlock Holmes

Penjelasan

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

Berfungsi untuk mendefinisikan struktur data yang disebut Node, memiliki dua anggota yaitu link dan alphabet.

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

Berfungsi untuk membuat variabel-variabel dari tipe data Node.

```
 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";
```

Berfungsi untuk membuat 9 node yang masing-masing memiliki pointer link yang awalnya diatur menjadi NULL dan pointer alphabet yang mengarah ke huruf tertentu.

```
 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;
```

Berfungsi untuk menghubungkan rangkaian batu yang ada di gambar menjadi sebuah linked list dengan urutan yang spesifik.

```
 printf("%s", l3.link->link->link->alphabet); //Output: I
  printf("%s", l3.link->link->link->link->alphabet); //Output: N
  printf("%s", l3.link->link->link->link->link->alphabet); //Output: F
  printf("%s", l3.link->link->link->link->link->link->alphabet); //Output: O
  printf("%s", l3.link->link->alphabet); //Output: R
  printf("%s", l3.link->link->link->link->link->link->link->alphabet);
//Output: M
  printf("%s", l3.alphabet); //Output: A
  printf("%s", l3.link->alphabet); //Output: T
  printf("%s", l3.link->link->link->alphabet); //Output: I
  printf("%s", l3.link->link->link->link->link->link->link->link->alphabet);
//Output: K
  printf("%s", l3.alphabet); //Output: A

  return 0;
}
```

Berfungsi untuk mencetak kata INFORMATIKA menggunakan simpul l3 sebagai titik awal.

Output:

```
PS C:\Users\RICHIE\Documents\alpro\coding\semester 2> cd "c:\Users\RICHIE\Documents\alpro\codi
ng\semester 2\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tem
pCodeRunnerFile }
INFORMATIKA
```

2. hackerrank

Penjelasan

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

Berfungsi untuk menghitung jumlah maksimum elemen yang dapat diambil dari kedua stack.

```
    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i];
        i++;
        count++;
    }
```

Berfungsi untuk mengambil elemen dari stack a selama nilai total elemen yang telah diambil sum tidak melebihi maxsum.

```
int maxCount = count;
```

Berfungsi untuk memberikan nilai awal pada suatu variable maxcount dengan nilai dari count.

```
while (j < m && i >= 0) {
        // Jika tambahan elemen dari stack b tidak memperbesar jumlah,
        // hapus elemen dari stack a dan update jumlah
        if (sum + b[j] > maxSum && i > 0) {
            i--;
            sum -= a[i];
            count--;
            continue;
        }
```

Berfungsi untuk mengoptimalkan jumlah elemen yang dapat diambil dari kedua stack.

```
 sum += b[j];
        j++;
        count++;
```

Berfungsi untuk menambahkan elemen dari stack b ke sum dan mengupdate count serta indeks j.

```
if (count > maxCount) {
            maxCount = count;
        }
```

berfungsi untuk memperbarui max count

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

berfungsi untuk memasukan jumlah kasus permainan yang akan dihitung.

```
 for (int i = 0; i < g; i++) {
```

Berfungsi untuk Melakukan iterasi sebanyak g.

```
 int n, m, maxSum;
```

```
        scanf("%d %d %d", &n, &m, &maxSum);
```
Berfungsi untuk membaca nilai n, m, dan maxum untuk kasus permainan saat ini.

```
int a[n], b[m];
```
Berfungsi untuk Mendeklarasikan array a dan b.

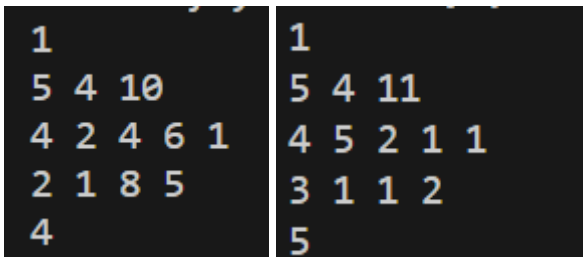
```
for (int j = 0; j < n; j++) {
            scanf("%d", &a[j]);
        }
```
Berfungsi untuk membaca nilai elemen-elemen stack a dan menyimpannya dalam array a.

```
    for (int j = 0; j < m; j++) {
            scanf("%d", &b[j]);
        }
```
Berfungsi untuk membaca nilai elemen-elemen stack b dan menyimpannya dalam array b.

```
int result = twoStacks(maxSum, n, m, a, b);
```
Berfungsi untuk memanggil fungsi twostacks untuk menghitung jumlah maksimum elemen yang dapat diambil.

```
printf("%d\n", result);
    }
```
Mencetak hasil perhitungan jumlah maksimum elemen yang dapat diambil untuk kasus permainan saat ini.

Output



Visualisasi

Langkah 1:

Stack a: [4, 2, 4, 6, 1] (top)

Stack b: [2, 1, 8, 5] (top)

Total elemen yang dapat diambil: 0

Langkah 2:

Ambil elemen 4 dari stack a

Stack a: [2, 4, 6, 1] (top)

Stack b: [2, 1, 8, 5] (top)

Total elemen yang dapat diambil: 1

Langkah 3:

Ambil elemen 2 dari stack a

Stack a: [4, 6, 1] (top)

Stack b: [2, 1, 8, 5] (top)

Total elemen yang dapat diambil: 2

Langkah 4:

Ambil elemen 4 dari stack a

Stack a: [6, 1] (top)

Stack b: [2, 1, 8, 5] (top)

Total elemen yang dapat diambil: 3

Langkah 5:

Ambil elemen 6 dari stack a

Stack a: [1] (top)

Stack b: [2, 1, 8, 5] (top)

Total elemen yang dapat diambil: 4