NAMA : NUR AULIA DINDA PUTRI

KELAS : IF-03-02

NIM : 1203230093

1.)

A.) SOURCE CODE

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

// Definisikan struktur batu
struct Stone {
    struct Stone* link;
    char* alphabet;
};

int main() {
    // Inisialisasi batu-batu
    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    // Inisialisasi huruf pada masing-masing batu
    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;
```

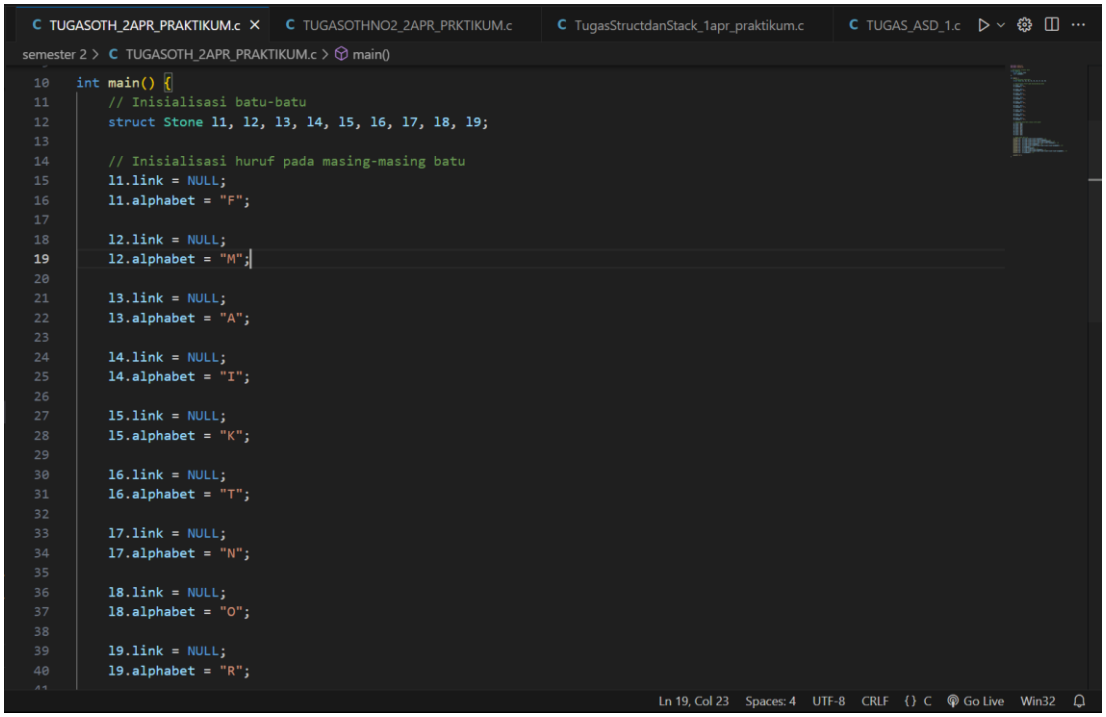
```
    l8.alphabet = "O";

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

    // Hubungkan batu-batu sesuai arah panah
    l3.link = &l6;
    l6.link = &l9;
    l9.link = &l4;
    l4.link = &l7;
    l7.link = &l1;
    l1.link = &l8;
    l8.link = &l2;
    l2.link = &l5;
    l5.link = &l3;

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

    printf("\n");
}
```

B.) OUTPUT

C.) PENJELASAN

```c
// Definisikan struktur batu
struct Stone {
    struct Stone* link;
    char* alphabet;
};
```

Struct stone untuk mendefinisikan sebuah struktur data baru (stone adalah nama struktur datanya). **Struct stone* link** untuk mendefinisikan elemen pertama bernama link dengan tipe data pointer,digunakan untuk menghubungkan struktur stone satu sama lain. **Char* alphabet** untuk mendefinisikan anggota kedua dari struktur stone,yg mrupakan pointer ke karakter ,digunakan untuk menhyimpan informasi tentang karakter dengan struktur stone ini.



struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9 Mendeklarasikan dan menginisialisasi sembilan variabel bertipe struct Stone dengan nama l1 hingga l9**. l1.link = NULL**; Mengatur pointer link pada batu l1 ke NULL, menunjukkan batu ini tidak terhubung dengan batu lainnya. **l1.alphabet = "F"** Mengatur pointer alphabet pada batu l1 ke alamat string "F".Perintah serupa diulang untuk batu l2 hingga l9, menginisialisasi pointer link dan alphabet dengan nilai yang sesuai. Jadi, secara keseluruhan, kode ini menginisialisasi sembilan batu dengan huruf-huruf yang berbeda dan mengatur koneksi antara batu-batu tersebut.

```c
// Hubungkan batu-batu sesuai arah panah
l3.link = &l6;
l6.link = &l9;
l9.link = &l4;
l4.link = &l7;
l7.link = &l1;
l1.link = &l8;
l8.link = &l2;
l2.link = &l5;
l5.link = &l3;
```

l3.link = &l6; Baris ini mengatur link dari batu ketiga (l3) untuk menunjuk ke alamat dari batu keenam (l6), menunjukkan bahwa batu ketiga terhubung ke batu keenam**. l6.link = &l9;** Baris ini mengatur link dari batu keenam (l6) untuk menunjuk ke alamat dari batu kesembilan (l9**). l9.link = &l4;** Baris ini mengatur link dari batu kesembilan (l9) untuk menunjuk ke alamat dari batu keempat (l4). **l4.link = &l7;** Baris ini mengatur link dari batu keempat (l4) untuk menunjuk ke alamat dari batu ketujuh (l7**). l7.link = &l1;** Baris ini mengatur link dari batu ketujuh (l7) untuk menunjuk ke alamat dari batu pertama (l1). **l1.link = &l8;** Baris ini mengatur link dari batu pertama (l1) untuk menunjuk ke alamat dari batu kedelapan (l8). **l8.link = &l2;** Baris ini mengatur link dari batu kedelapan (l8) untuk menunjuk ke alamat dari batu kedua (l2). **l2.link = &l5;** Baris ini mengatur link dari batu kedua (l2) untuk menunjuk ke alamat dari batu kelima (l5). **l5.link = &l3;** Baris ini mengatur link dari batu kelima (l5) untuk menunjuk ke alamat dari batu ketiga (l3).

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

    printf("\n");
}
```

Blok kode ini mendemonstrasikan bagaimana mengakses karakter yang disimpan dalam linked list dengan mengikuti pointer link dari batu awal (l3) hingga mencapai batu yang diinginkan. Perintah printf kemudian mencetak karakter tersebut. Proses pencetakan dilakukan dengan mengikuti langkah-langkah melalui pointer link dari l3 menuju batu-batu terhubung lainnya dalam urutan yang telah ditentukan sebelumnya. Melalui penggunaan pointer dan akses ke data yang terkait dengan setiap struktur Stone, kode tersebut memperlihatkan penggunaan linked list dalam menyimpan dan mengakses data secara terstruktur.

2.)
A.) SOURCE CODE

```c
#include <stdio.h>

int twoStacks(int maxSum, int a[], int n, int b[], int m) { //deklarasi fungsi dg nama twistacks yg menerima enam paramenter
    int sum = 0, count = 0, temp = 0, i = 0, j = 0; //untuk mendeklarasi bbrpa variabel yg akn digunakan dalam fungsi,sum untuk myimpna jumlah elemen dr stack,count untuk mnyimpn jmlh total elemen yg bs diambil dri kedua tumpukn,temp untuk penyimpnan sementra

    while (i < n && sum + a[i] <= maxSum) { //loop ini akn trs berjalan selama i < n,Jumlah saat ini (sum) ditambah elemen pada indeks i dalam a tidak melebihi batas maxSum.
        sum += a[i++]; // Elemen pada a[i] ditambahkan ke sum, dan i dinaikkan untuk pindah ke elemen berikutnya di a.
    }
    count = i; //nilai i ditetapkan ke count. Ini adalah nilai awal untuk jumlah elemen maksimum.

    while (j < m && i >= 0) {  //Looping kedua ini akan terus berjalan selama j < m dan tidak i < dari 0
        sum += b[j++]; // Menambahkan nilai b[j] ke sum dan meningkatkan j setiap kali loop berjalan.
        while (sum > maxSum && i > 0) { // looping nested. Looping ini akan terus berjalan selama sum lebih besar dari maxSum
            sum -= a[--i]; // ini akan mengurangi nilai a[i] dari sum dan mengurangi i setiap kali loop berjalan.
        }
        if (sum <= maxSum && i + j > count) { //Jika sum kurang dari atau sama dengan maxSum dan jumlah total elemen (i + j) lebih besar dari count
            count = i + j; //maka count akan diperbarui dengan nilai i + j.
        }
    }
    return count; //Mengembalikan nilai count setelah selesai menjalankan semua operasi di dalam fungsi.
}

int main() {
    int g; // Mendeklarasikan variabel g yang akan digunakan untuk menyimpan jumlah kasus uji.
    scanf("%d", &g);
```

```c
    while (g--) { //Memulai loop while yang akan berjalan sebanyak g kali, yaitu sesuai dengan jumlah kasus uji yang diinputkan sebelumnya.
        int n, m, maxSum; //Mendeklarasikan variabel n, m, dan maxSum yang akan digunakan untuk menyimpan ukuran dua tumpukan (array) dan jumlah maksimum yang dapat ditambahkan dari kedua tumpukan tersebut.
        scanf("%d%d%d", &n, &m, &maxSum);
        int a[n], b[m]; // Mendeklarasikan dua array a dan b dengan ukuran sesuai dengan nilai n dan m yang telah diinputkan
        for (int i = 0; i < n; i++) { // Loop for untuk membaca nilai-nilai elemen dari tumpukan pertama (a) dan menyimpannya dalam array a.
            scanf("%d", &a[i]);
        }
        for (int i = 0; i < m; i++) { //Loop for untuk membaca nilai-nilai elemen dari tumpukan kedua
            scanf("%d", &b[i]);
        }
        printf("%d\n", twoStacks(maxSum, a, n, b, m)); //Memanggil fungsi twoStacks dengan argumen jumlah maksimum yang dapat ditambahkan (maxSum),array tumpukan pertama (a), ukuran tumpukan pertama (n), array tumpukan kedua (b), dan ukuran tumpukan kedua (m). Hasil dari pemanggilan ini kemudian dicetak.
    }
    return 0; //Mengembalikan nilai 0 yang menandakan bahwa program telah berakhir dengan sukses.
}
```

B.) OUTPUT