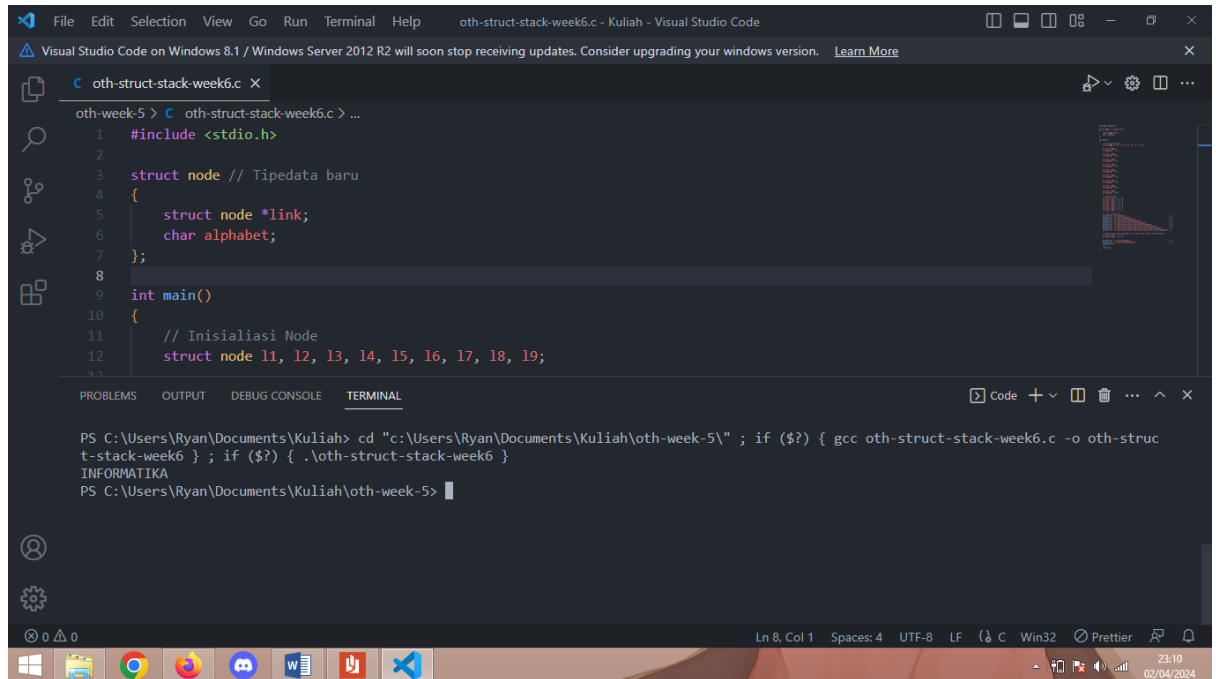


Teguh Ryan Firmansyah

1203230044

1. Asisten Sherlock Holmes

Output



The screenshot shows the Visual Studio Code interface. The editor displays a C program named `oth-struct-stack-week6.c`. The code defines a `struct node` with a pointer to another `struct node` and a character `alphabet`. The `main` function initializes an array of 19 `struct node` objects. The terminal window shows the command to compile the program using `gcc` and the resulting executable `oth-struct-stack-week6.exe`.

```
1 #include <stdio.h>
2
3 struct node // Tipedata baru
4 {
5     struct node *link;
6     char alphabet;
7 };
8
9 int main()
10 {
11     // Inisialisasi Node
12     struct node 11, 12, 13, 14, 15, 16, 17, 18, 19;
```

```
PS C:\Users\Ryan\Documents\Kuliah> cd "c:\Users\Ryan\Documents\Kuliah\oth-week-5\" ; if ($?) { gcc oth-struct-stack-week6.c -o oth-struct-stack-week6.exe } ; if ($?) { .\oth-struct-stack-week6.exe }
INFORMATIKA
PS C:\Users\Ryan\Documents\Kuliah\oth-week-5>
```

Penjelasan Code

```
#include <stdio.h>

struct node // Tipedata baru
{
    struct node *link;
    char alphabet;
};
```

Membuat Tipedata baru bernama node yang berisikan struct node itu sendiri(memanggil typedata itu sendiri) yang dinamai link yang merupakan pointer agar merujuk ke address dan char alphabet

```

int main()
{
    // Inisialisasi Node
    struct node 11, 12, 13, 14, 15, 16, 17, 18, 19;

    11.link = NULL;
    11.alphabet = 'F';

    12.link = NULL;
    12.alphabet = 'M';

    13.link = NULL;
    13.alphabet = 'A';

    14.link = NULL;
    14.alphabet = 'I';

    15.link = NULL;
    15.alphabet = 'K';

    16.link = NULL;
    16.alphabet = 'T';

    17.link = NULL;
    17.alphabet = 'N';

    18.link = NULL;
    18.alphabet = 'O';

    19.link = NULL;
    19.alphabet = 'R';
}

```

Di int main() struct node untuk menginisialisasi typedata yang tadi telah kita buat menggunakan struct bernama node, dengan variable 11,12-19.

Lalu tiap variabel / angka tadi disambungkan dengan link dan juga alphabet,ingat lan di dalam typedata node ada dua variabel dengan beda typedata yakni struct node itu sendiri(nested) dan char alphabet.

Untuk sambungan link di isi NULL agar tidak mengubah address,lalu alphabet di isikan huruf yang nantinya menjadi I N F O R M A T I K A (sesuai soal aja sih)

```

// Linking nodes
14.link = &17; // I > N
17.link = &11; // N > F
11.link = &18; // F > O
18.link = &19; // O > R
19.link = &12; // R > M
12.link = &13; // M > A
13.link = &16; // A > T
16.link = &14; // T > I

```

Linked node di sini guna untuk menyambungkan dari alphabet satu ke alphabet selanjutnya dan diberi & dikarenakan ini merujuk ke address / ram

```

// Print linked list
printf("%c", 14.alphabet); // I
printf("%c", 14.link->alphabet); // N
printf("%c", 14.link->link->alphabet); // F
printf("%c", 14.link->link->link->alphabet); // O
printf("%c", 14.link->link->link->link->alphabet); // R
printf("%c", 14.link->link->link->link->link->alphabet); // M
printf("%c", 14.link->link->link->link->link->link->alphabet); // A
printf("%c", 14.link->link->link->link->link->link->link->alphabet); // T
printf("%c", 14.link->link->link->link->link->link->link->link->alphabet); // I

// Dipisah agar tidak mengubah nilai yang telah dibuat terlebih dahulu
14.link = &15; // I > K
15.link = &13; // K > A

printf("%c", 14.link->alphabet); // K
printf("%c", 14.link->link->alphabet); // A
printf("\n");

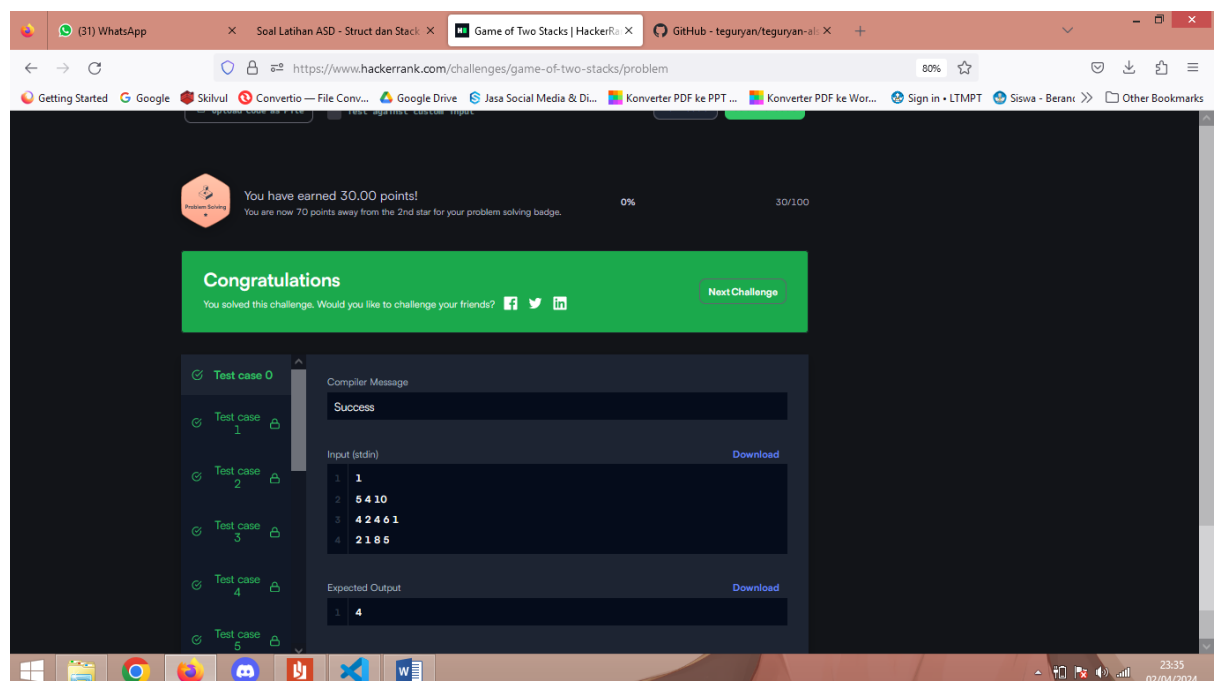
return 0;
}

```

Lalu sprint linked list untuk print perhurufnya. Dikarenakan i huruf pertama maka langsung aja 14.alphabet agar merujuk ke i,lalu setelahnya bisa diberi link sebelum alphabet dan dikalilipatkan saja sampai INFORMATI

Dan dipisah dikarenakan huruf i telah dijalankan program,jika langsung dan tidak dipisah,maka program akan mengubah nilai keseluruhan,maka dari itu dipisah (untuk i pertama dan i setelah T agar tidak berubah nilainya) setelah itu diprint lagi

2. Hackerrank



```

int twoStacks(int maxSum, int a_count, int *a, int b_count, int *b)
{
    int i = 0, j = 0, sum = 0, count = 0;
    while (i < a_count && sum + a[i] <= maxSum)
    {
        sum += a[i];
        i++;
    }
    count = i;
    while (j < b_count && i >= 0)
    {
        sum += b[j];
        j++;
        while (sum > maxSum && i > 0)
        {
            i--;
            sum -= a[i];
        }
        if (sum <= maxSum && i + j > count)
        {
            count = i + j;
        }
    }
    return count;
}

```

- Int TwoStacks dengan parameter int MaxSum,a_count,*a,b_count,*b
- Inisialisasi i,j aum,dan count Noya bernilai 0 agar pengulangan berikutnya tidak perlu inisialisasi
- Pengulangan menggunakan while agar mengloop melacak berapa banyak elemen yang diambil dari stack pertama tanpa melebihi maxSum
- Inisialisasi count = i agar mewakili jumlah Max elemen yang dapat diambil dari stack pertama sejauh ini
- Pengulangan while kedua agar kode menambahkan elemen saat ini dari stack kedua ke sum dan menghitung penghitung j . selama sum melebihi max Sum dan ada elemen distack pertama(i lebih besar dari 0).
- Di dalam while kedua ada pengulangan while lagi. Kode ini menghapus elemen terakhir dari stack pertama dan mengurangi nilainya dari sum.Ini memastikan bahwa sum tidak melebihi maxSum.
- Lalu ada If lagi jika diperlukan agar menyesuaikan elemen dari stack pertama, kode ini memeriksa apakah sum saat ini kurang dari atau sama dengan maxSum dan jumlah gabungan elemen dari kedua stack(i+j) lebihbesar dari count saat ini.
Jika kedua kondisi benar,kode ini memperbarui count kejumlah elemen gabungan(i+j). Ini memastikan bahwa count mewakili jumlah maksimum elemen yang dapat diambil dari kedua stack tanpa melebihi maxSum.
- Lalu mereturn count karena ini function / fungsi jadi harus mengembalikan satu nilai jika fungsi ini dipanggil