LAPORAN TUGAS OTH 1 & 2 PRAKTIKUM ASD

NAMA : Fandi Ardiansyah

KELAS : IF-03-03

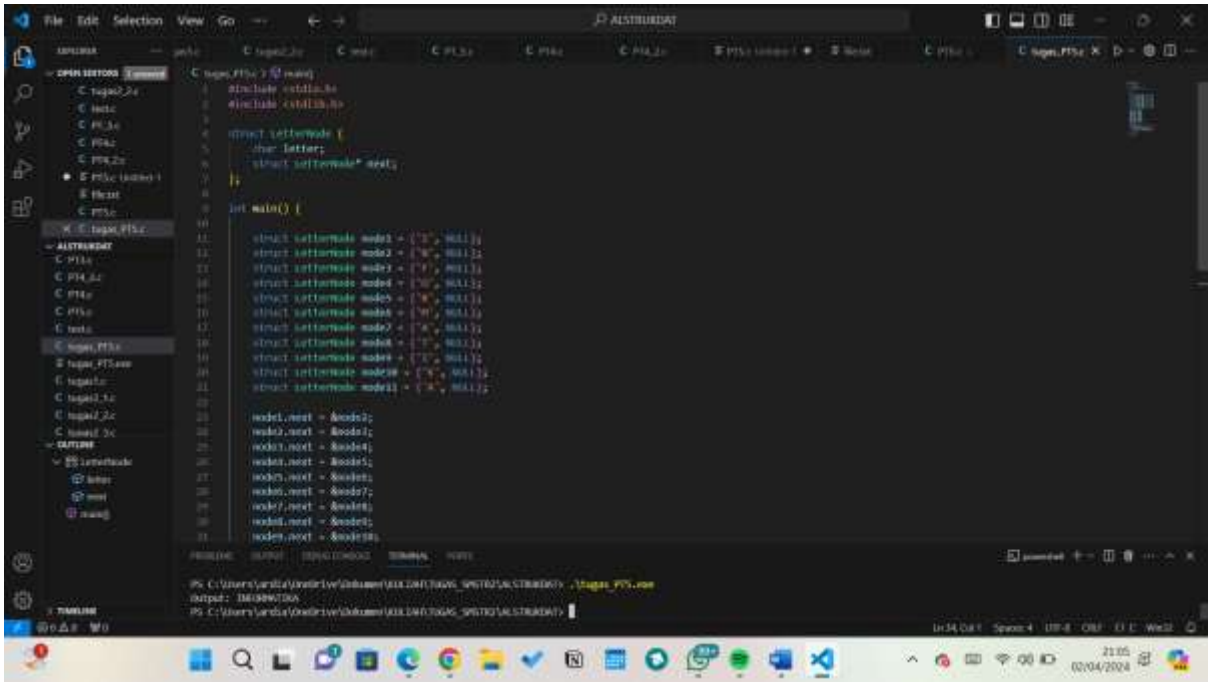
NIM : 1203230079

1. **Source Code**

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

struct LetterNode {
    char letter;
    struct LetterNode* next;
};

int main() {

    struct LetterNode node1 = {'I', NULL};
    struct LetterNode node2 = {'N', NULL};
    struct LetterNode node3 = {'F', NULL};
    struct LetterNode node4 = {'O', NULL};
    struct LetterNode node5 = {'R', NULL};
    struct LetterNode node6 = {'M', NULL};
    struct LetterNode node7 = {'A', NULL};
    struct LetterNode node8 = {'T', NULL};
    struct LetterNode node9 = {'I', NULL};
    struct LetterNode node10 = {'K', NULL};
    struct LetterNode node11 = {'A', NULL};

    node1.next = &node2;
    node2.next = &node3;
    node3.next = &node4;
    node4.next = &node5;
    node5.next = &node6;
    node6.next = &node7;
    node7.next = &node8;
    node8.next = &node9;
    node9.next = &node10;
    node10.next = &node11;
    node11.next = NULL;

    printf("Output: ");
    struct LetterNode* currentNode = &node1;
    while (currentNode != NULL) {
        printf("%c", currentNode->letter);
```

```
        currentNode = currentNode->next;
    }
    printf("\n");

    return 0;
}
```

Output



Penjelasan

1. `#include <stdio.h>`: Mengimpor header file standar untuk operasi input/output.
2. `#include <stdlib.h>`: Mengimpor header file standar untuk fungsi dan tipe data yang diperlukan.

3. `struct LetterNode {`: Mendefinisikan struktur `LetterNode` yang akan digunakan untuk merepresentasikan setiap node dalam linked list.
4. `char letter;`: Mendefinisikan variabel `letter` dalam struktur untuk menyimpan karakter huruf.
5. `struct LetterNode* next;`: Mendefinisikan pointer `next` dalam struktur untuk menunjuk ke node berikutnya dalam linked list.
6. `};`: Menutup definisi struktur `LetterNode`.

7. `int main() {`: Mendefinisikan fungsi utama program.

8. `struct LetterNode node1 = {'I', NULL};`: Membuat node pertama dengan huruf 'I' dan pointer `next` diatur ke `NULL`.

9. `struct LetterNode node2 = {'N', NULL};`: Membuat node kedua dengan huruf 'N' dan pointer `next` diatur ke `NULL`.
10. `struct LetterNode node3 = {'F', NULL};`: Membuat node ketiga dengan huruf 'F' dan pointer `next` diatur ke `NULL`.
11. `struct LetterNode node4 = {'O', NULL};`: Membuat node keempat dengan huruf 'O' dan pointer `next` diatur ke `NULL`.
12. `struct LetterNode node5 = {'R', NULL};`: Membuat node kelima dengan huruf 'R' dan pointer `next` diatur ke `NULL`.
13. `struct LetterNode node6 = {'M', NULL};`: Membuat node keenam dengan huruf 'M' dan pointer `next` diatur ke `NULL`.
14. `struct LetterNode node7 = {'A', NULL};`: Membuat node ketujuh dengan huruf 'A' dan pointer `next` diatur ke `NULL`.
15. `struct LetterNode node8 = {'T', NULL};`: Membuat node kedelapan dengan huruf 'T' dan pointer `next` diatur ke `NULL`.
16. `struct LetterNode node9 = {'I', NULL};`: Membuat node kesembilan dengan huruf 'I' dan pointer `next` diatur ke `NULL`.
17. `struct LetterNode node10 = {'K', NULL};`: Membuat node kesepuluh dengan huruf 'K' dan pointer `next` diatur ke `NULL`.
18. `struct LetterNode node11 = {'A', NULL};`: Membuat node kesebelas dengan huruf 'A' dan pointer `next` diatur ke `NULL`.

19. `node1.next = &node2;`: Menghubungkan node pertama (`node1`) dengan node kedua (`node2`) dalam linked list.
20. `node2.next = &node3;`: Menghubungkan node kedua (`node2`) dengan node ketiga (`node3`) dalam linked list.
21. `node3.next = &node4;`: Menghubungkan node ketiga (`node3`) dengan node keempat (`node4`) dalam linked list.
22. `node4.next = &node5;`: Menghubungkan node keempat (`node4`) dengan node kelima (`node5`) dalam linked list.
23. `node5.next = &node6;`: Menghubungkan node kelima (`node5`) dengan node keenam (`node6`) dalam linked list.
24. `node6.next = &node7;`: Menghubungkan node keenam (`node6`) dengan node ketujuh (`node7`) dalam linked list.
25. `node7.next = &node8;`: Menghubungkan node ketujuh (`node7`) dengan node kedelapan (`node8`) dalam linked list.
26. `node8.next = &node9;`: Menghubungkan node kedelapan (`node8`) dengan node kesembilan (`node9`) dalam linked list.
27. `node9.next = &node10;`: Menghubungkan node kesembilan (`node9`) dengan node kesepuluh (`node10`) dalam linked list.
28. `node10.next = &node11;`: Menghubungkan node kesepuluh (`node10`) dengan node kesebelas (`node11`) dalam linked list.
29. `node11.next = NULL;`: Menandai akhir dari linked list dengan mengatur pointer `next` dari node kesebelas (`node11`) menjadi `NULL`.

30. `printf("Output: ");`: Mencetak string "Output: " ke layar.

31. `struct LetterNode* currentNode = &node1;`: Menginisialisasi pointer `currentNode` untuk menunjuk ke node pertama (`node1`) sebagai titik awal untuk iterasi.

32. `while (currentNode != NULL) {`: Mulai loop `while` yang akan berjalan selama `currentNode` tidak `NULL`, yang berarti kita belum mencapai akhir dari linked list.

33. `printf("%c", currentNode->letter);`: Mencetak karakter huruf yang disimpan dalam node yang ditunjuk oleh `currentNode`.

34. `currentNode = currentNode->next;`: Mengubah `currentNode` untuk menunjuk ke node berikutnya dalam linked list.

35. `}`: Menutup loop `while`.

36. `printf("\n");`: Mencetak newline ke layar untuk memisahkan output dari kode berikutnya.

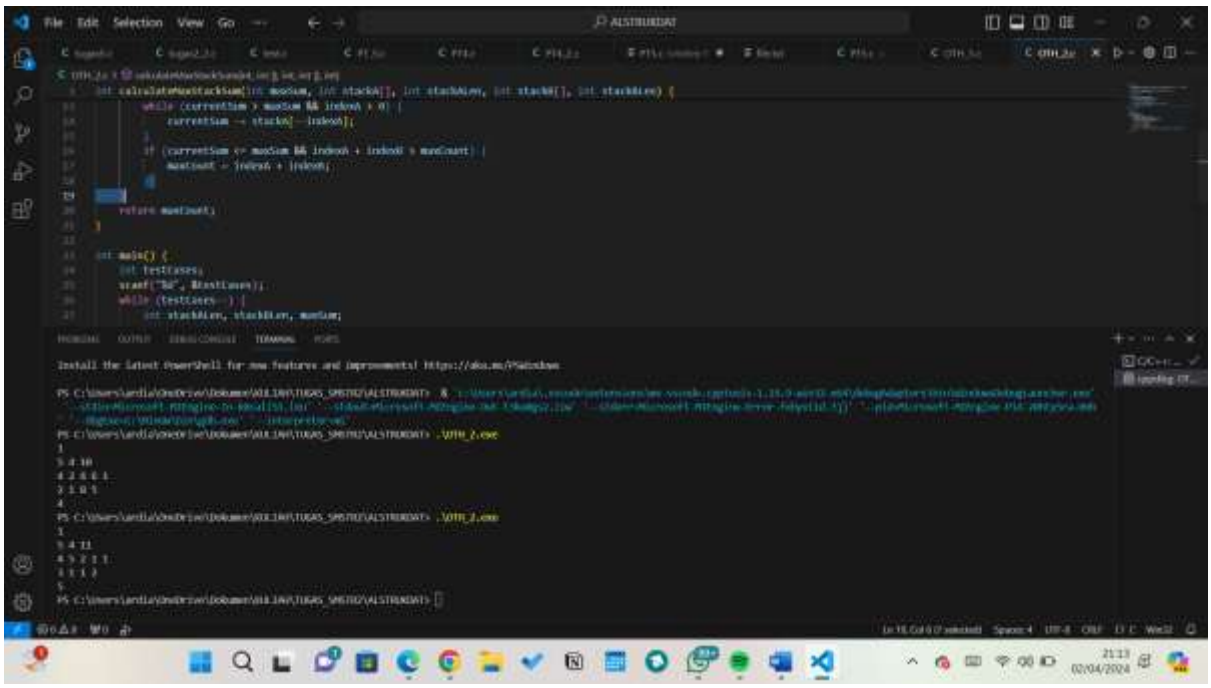
37. `return 0;`: Mengembalikan nilai 0 dari fungsi `main`, menandai akhir dari program.

38. `}`: Menutup fungsi `main`.

2. **Source Code**

```
#include <stdio.h>

int calculateMaxStackSum(int maxSum, int stackA[], int stackALen, int
stackB[], int stackBLen) {
    int currentSum = 0, maxCount = 0, temp = 0, indexA = 0, indexB = 0;

    while (indexA < stackALen && currentSum + stackA[indexA] <= maxSum) {
        currentSum += stackA[indexA++];
    }
    maxCount = indexA;

    while (indexB < stackBLen && indexA >= 0) {
        currentSum += stackB[indexB++];
        while (currentSum > maxSum && indexA > 0) {
            currentSum -= stackA[--indexA];
        }
        if (currentSum <= maxSum && indexA + indexB > maxCount) {
            maxCount = indexA + indexB;
        }
```

```
    }
    return maxCount;
}

int main() {
    int testCases;
    scanf("%d", &testCases);
    while (testCases--) {
        int stackALen, stackBLen, maxSum;
        scanf("%d%d%d", &stackALen, &stackBLen, &maxSum);
        int stackA[stackALen], stackB[stackBLen];
        for (int i = 0; i < stackALen; i++) {
            scanf("%d", &stackA[i]);
        }
        for (int i = 0; i < stackBLen; i++) {
            scanf("%d", &stackB[i]);
        }
        printf("%d\n", calculateMaxStackSum(maxSum, stackA, stackALen, stackB,
stackBLen));
    }
    return 0;
}
```



Penjelasan

1. Pendefinisian Header: `#include <stdio.h>` mengimpor header file standar untuk operasi input/output.
2. Fungsi `calculateMaxStackSum`: Fungsi ini menerima `maxSum`, dua array `stackA` dan `stackB`, serta panjang masing-masing array. Fungsi ini mengembalikan jumlah maksimum elemen yang dapat disatukan tanpa melebihi `maxSum`.
3. Inisialisasi Variabel: Di dalam fungsi `calculateMaxStackSum`, beberapa variabel diinisialisasi untuk menghitung jumlah elemen dan sum dari elemen-elemen dalam stack.
4. Loop Pertama: Loop ini mengambil elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Jika sum melebihi `maxSum`, loop ini berhenti dan `maxCount` diatur ke jumlah elemen yang telah diambil.
5. Loop Kedua: Loop ini mengambil elemen dari `stackB` dan menambahkannya ke `currentSum`. Jika `currentSum` melebihi `maxSum`, loop ini mengurangi elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Jika sum tidak melebihi `maxSum` dan jumlah elemen saat ini lebih besar dari `maxCount`, `maxCount` diperbarui.
6. Fungsi `main`: Fungsi ini mengatur alur program, termasuk membaca input dan memanggil `calculateMaxStackSum`.
7. Membaca Input: Program membaca jumlah kasus uji, panjang stack, dan `maxSum`. Kemudian membaca elemen-elemen dari kedua stack.
8. Mencetak Hasil: Program mencetak hasil dari `calculateMaxStackSum` untuk setiap kasus uji.

Visualisasi Alur Penyelesaian dengan Input:

Kasus Uji :

- `maxSum` = 11
- `stackA` = [4, 5, 2, 1, 1]
- `stackB` = [3, 1, 1, 2]

Alur Penyelesaian:

1. Mulai dengan `stackA`, ambil elemen sampai sum tidak melebihi `maxSum`. Dalam hal ini, kita ambil elemen 4 dan 5 dari `stackA`, sehingga sum = 9.
2. Kemudian, mulai dengan `stackB`, tambahkan elemen sampai sum tidak melebihi `maxSum`. Kita tambahkan elemen 3 dari `stackB`, sehingga sum = 12. Karena sum melebihi `maxSum`, kita mengurangi elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Kita mengurangi elemen 5 dari `stackA`, sehingga sum = 9.
3. Lanjutkan dengan `stackB`, tambahkan elemen 1 dan 1, sehingga sum = 11. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 2, sehingga sum = 13. Karena sum melebihi `maxSum`, kita mengurangi elemen

dari `stackA` sampai sum tidak melebihi `maxSum`. Kita mengurangi elemen 4 dari `stackA`, sehingga sum = 5.

4. Kemudian, kita tambahkan elemen 1 dan 1 dari `stackB`, sehingga sum = 7. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 2, sehingga sum = 9. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 3, sehingga sum = 12. Karena sum melebihi `maxSum`, kita mengurangi elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Kita mengurangi elemen 2 dari `stackA`, sehingga sum = 3.
5. Kemudian, kita tambahkan elemen 1 dari `stackB`, sehingga sum = 4. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 1, sehingga sum = 5. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 2, sehingga sum = 7. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 3, sehingga sum = 10. Karena sum melebihi `maxSum`, kita mengurangi elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Kita mengurangi elemen 1 dari `stackA`, sehingga sum = 3.
6. Kemudian, kita tambahkan elemen 1 dari `stackB`, sehingga sum = 4. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 1, sehingga sum = 5. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 2, sehingga sum = 7. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 3, sehingga sum = 10. Karena sum melebihi `maxSum`, kita mengurangi elemen dari `stackA` sampai sum tidak melebihi `maxSum`. Kita mengurangi elemen 1 dari `stackA`, sehingga sum = 3.
7. Kemudian, kita tambahkan elemen 1 dari `stackB`, sehingga sum = 4. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 1, sehingga sum = 5. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 2, sehingga sum = 7. Karena sum tidak melebihi `maxSum`, kita lanjutkan dengan `stackB` dan tambahkan elemen 3, sehingga sum = 10. Karena sum melebihi `maxSum