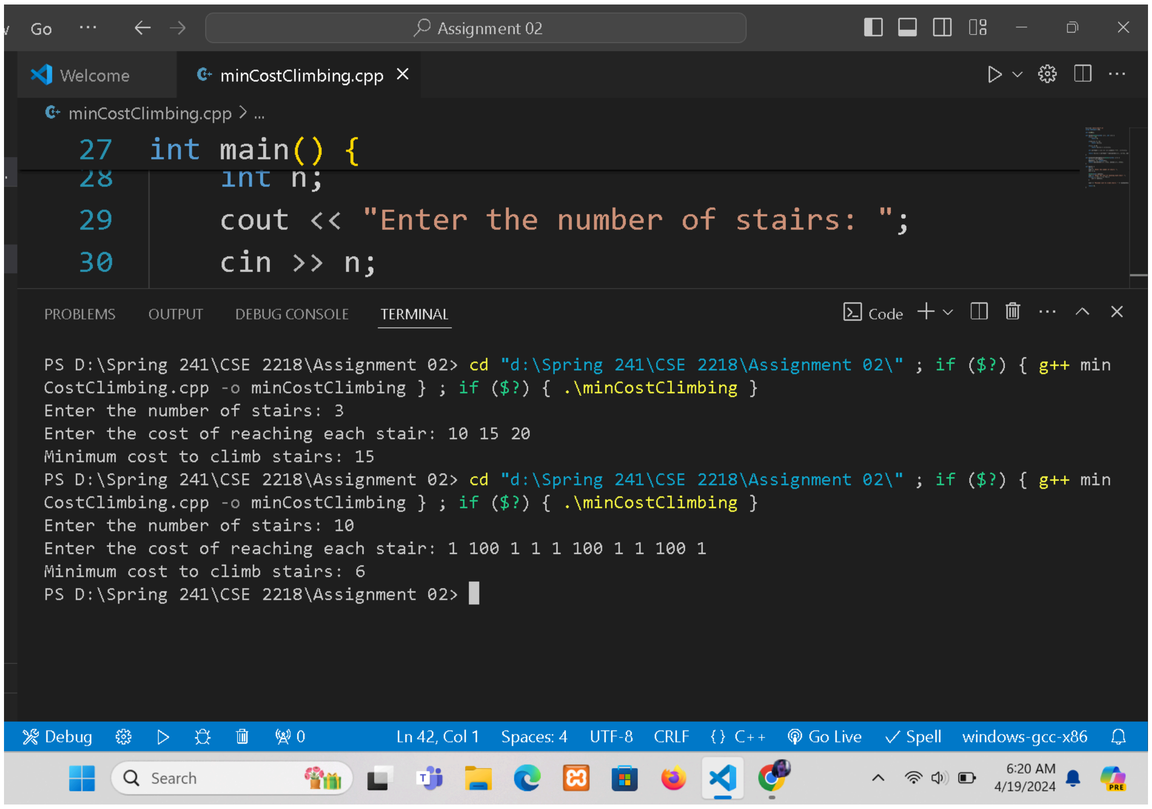
1. Minimum Cost for Climbing Stairs

```cpp
#include <bits/stdc++.h>
using namespace std;

int dp[1001];

int solve(vector<int>& cost, int idx) {
    if(idx < 0)
        return 0;

    if(dp[idx] != -1) return dp[idx];

    if(idx == 0) return dp[idx] = cost[idx];

    int currCost = (idx == cost.size()) ? 0 : cost[idx];

    return dp[idx] = currCost + min(solve(cost, idx-1), solve(cost, idx-2));
}
```

```cpp
int minCostClimbingStairs(vector<int>&
cost) {
    int n = cost.size();
    memset(dp, -1, sizeof(dp));
    return min(solve(cost, n-1),
solve(cost, n-2));
}

int main() {
    int n;
    cout << "Enter the number of
stairs: ";
    cin >> n;

    vector<int> cost(n);
    cout << "Enter the cost of reaching
each stair: ";
    for(int i = 0; i < n; ++i) {
        cin >> cost[i];
    }

    cout << "Minimum cost to climb
stairs: " <<
minCostClimbingStairs(cost) << endl;
```

```
    return 0;
}
```

Output:



```
int main() {
    int n;
    cout << "Enter the number of stairs: ";
    cin >> n;
```

```
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ min
CostClimbing.cpp -o minCostClimbing } ; if ($?) { .\minCostClimbing }
Enter the number of stairs: 3
Enter the cost of reaching each stair: 10 15 20
Minimum cost to climb stairs: 15
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ min
CostClimbing.cpp -o minCostClimbing } ; if ($?) { .\minCostClimbing }
Enter the number of stairs: 10
Enter the cost of reaching each stair: 1 100 1 1 1 100 1 1 100 1
Minimum cost to climb stairs: 6
PS D:\Spring 241\CSE 2218\Assignment 02>
```

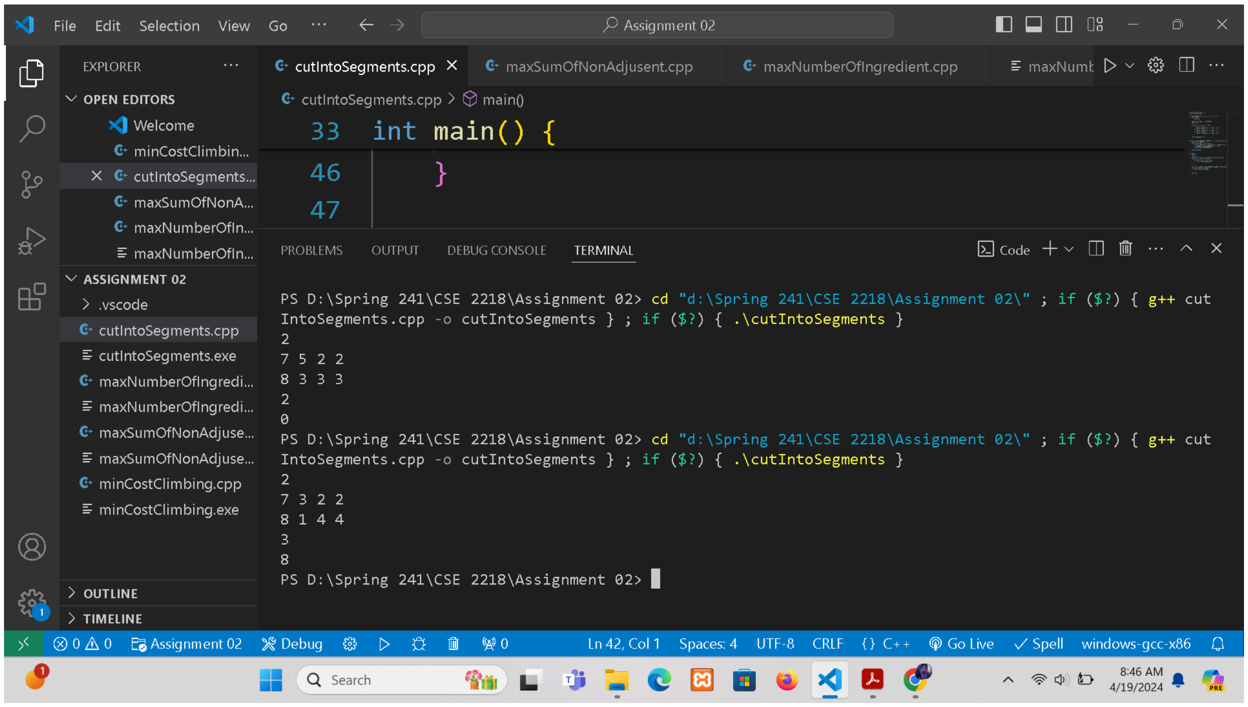
2. Cut Into three Segments

```
#include <bits/stdc++.h>
using namespace std;
```

```
int max_segments(int N, int X, int Y,
int Z) {
    if (N <= 0)
        return 0;

    vector<int> dp(N + 1, INT_MIN);
    dp[0] = 0;

    for (int i = 1; i <= N; ++i) {
        if (i - X >= 0)
            dp[i] = max(dp[i], 1 + dp[i
- X]);
        if (i - Y >= 0)
            dp[i] = max(dp[i], 1 + dp[i
- Y]);
        if (i - Z >= 0)
            dp[i] = max(dp[i], 1 + dp[i
- Z]);
    }

    return max(dp[N], 0);
}
```

```
vector<int> max_segments_wrapper(int T,
vector<vector<int>>& test_cases) {
    vector<int> results;
    for (auto& case_ : test_cases) {
        int N = case_[0], X = case_[1],
Y = case_[2], Z = case_[3];
        int segments = max_segments(N,
X, Y, Z);
        results.push_back(segments);
    }
    return results;
}

int main() {
    int T;
    cin >> T;
    vector<vector<int>> test_cases(T,
vector<int>(4));
    for (int i = 0; i < T; ++i) {
        for (int j = 0; j < 4; ++j) {
            cin >> test_cases[i][j];
        }
    }
    vector<int> output =
max_segments_wrapper(T, test_cases);
    for (int result : output) {
        cout << result << endl;
    }

    return 0;
}
```

Output:



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

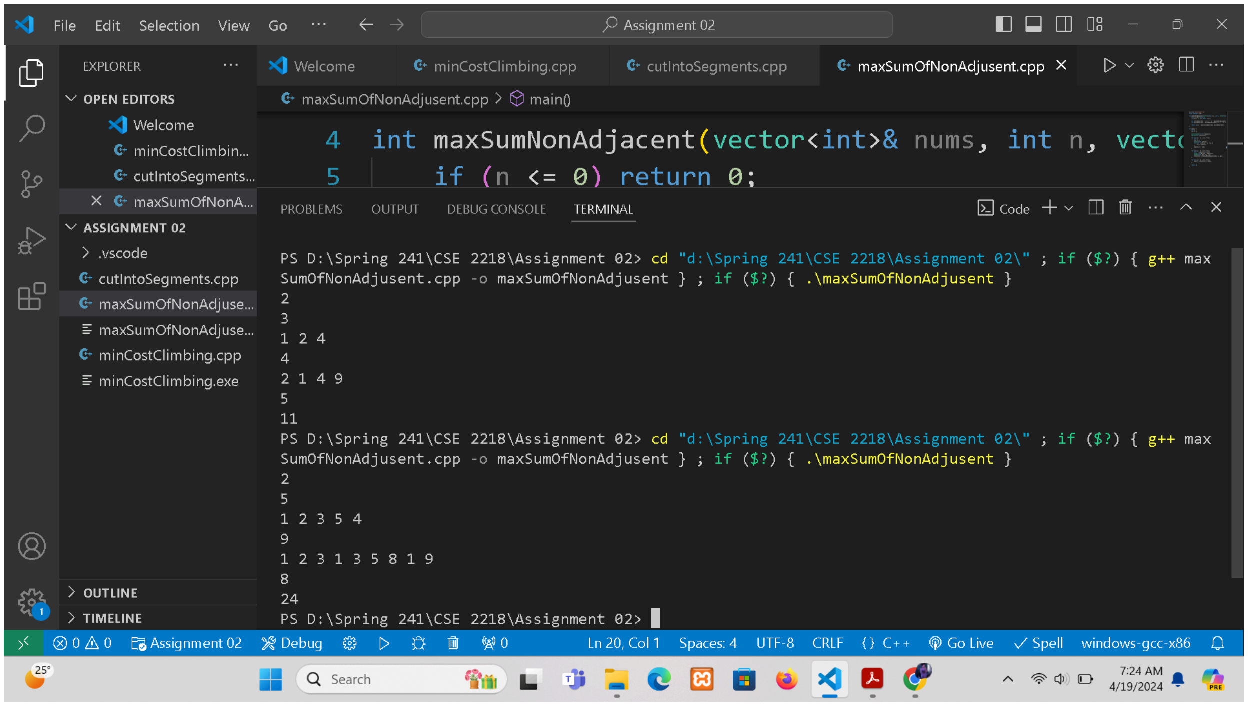
```
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ cut
IntoSegments.cpp -o cutIntoSegments } ; if ($?) { .\cutIntoSegments }
2
7 5 2 2
8 3 3 3
2
0
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ cut
IntoSegments.cpp -o cutIntoSegments } ; if ($?) { .\cutIntoSegments }
2
7 3 2 2
8 1 4 4
3
8
PS D:\Spring 241\CSE 2218\Assignment 02>
```

3. Maximum Sum of Non-Adjacent Elements

```
#include <bits/stdc++.h>
using namespace std;
```

```
int maxSumNonAdjacent(vector<int>& nums,
int n, vector<int>& dp) {
    if (n <= 0) return 0;
    if (dp[n] != -1) return dp[n];

    int includeCurrent = nums[n - 1] +
maxSumNonAdjacent(nums, n - 2, dp);
    int excludeCurrent =
maxSumNonAdjacent(nums, n - 1, dp);

    return dp[n] = max(includeCurrent,
excludeCurrent);
}

int main() {
    int t;
    cin >> t;

    vector<vector<int>> inputs(t);
    vector<int> results(t);

    for (int i = 0; i < t; ++i) {
        int n;
        cin >> n;
        vector<int> nums(n);
        for (int j = 0; j < n; ++j) {
            cin >> nums[j];
        }
        inputs[i] = nums;
    }

    for (int i = 0; i < t; ++i) {
        int n = inputs[i].size();
        vector<int>& nums = inputs[i];
        vector<int> dp(n + 1, -1);
        results[i] =
maxSumNonAdjacent(nums, n, dp);
    }

    for (int i = 0; i < t; ++i) {
        cout << results[i] << endl;
    }

    return 0;
}
```

Output:



```cpp
int maxSumNonAdjacent(vector<int>& nums, int n, vecto
    if (n <= 0) return 0;
```

```
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ max
SumOfNonAdjusent.cpp -o maxSumOfNonAdjusent } ; if ($?) { .\maxSumOfNonAdjusent }
2
3
1 2 4
4
2 1 4 9
5
11
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ max
SumOfNonAdjusent.cpp -o maxSumOfNonAdjusent } ; if ($?) { .\maxSumOfNonAdjusent }
2
5
1 2 3 5 4
9
1 2 3 1 3 5 8 1 9
8
24
PS D:\Spring 241\CSE 2218\Assignment 02>
```

4. TASFIA NEEDS TO BAKE A CAKE!!

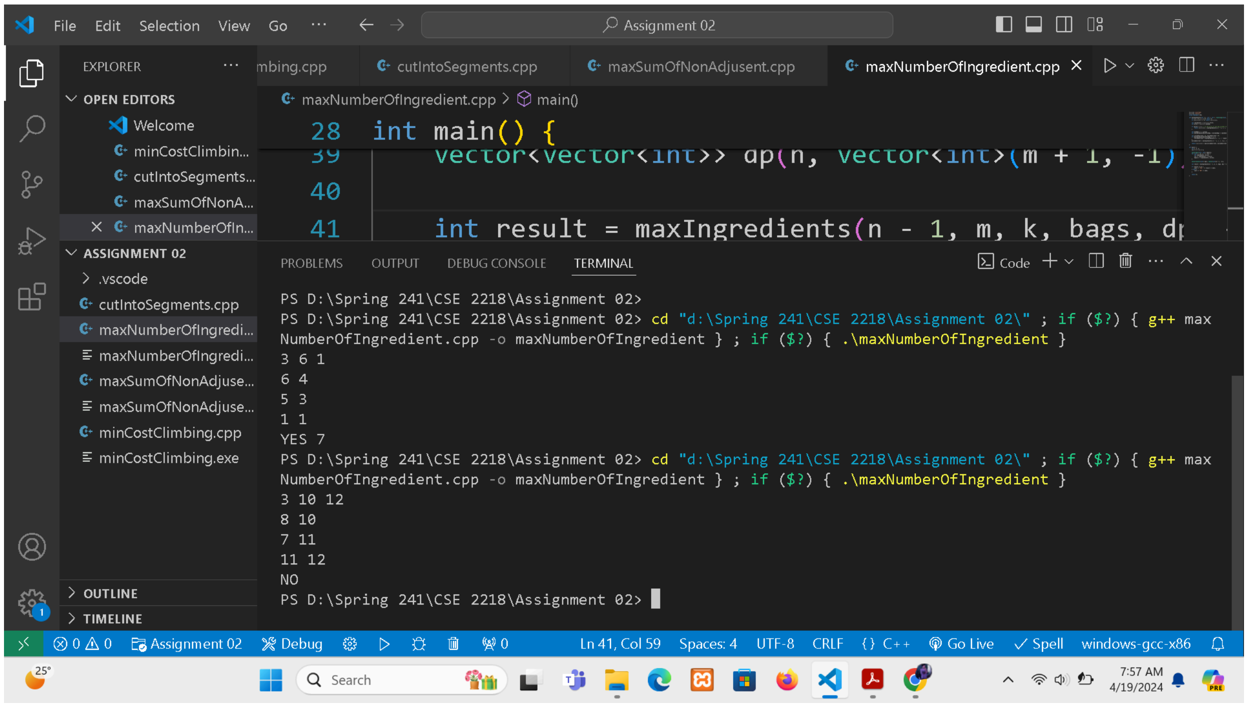
```cpp
#include <iostream>
#include <vector>
using namespace std;

int maxIngredients(int idx, int m, int k,
vector<pair<int, int>>& bags,
vector<vector<int>>& dp) {
    if (idx < 0 || m <= 0) return 0;
    if (dp[idx][m] != -1) return
dp[idx][m];
```

```
    int ingredients = bags[idx].first;
    int price = bags[idx].second;

    if (price > m) { // If the price of the
bag exceeds the available budget, exclude
it
        return dp[idx][m] =
maxIngredients(idx - 1, m, k, bags, dp);
    }

    int maxBags = m / price;
    int maxIngredientsWithThisBag =
min(maxBags * ingredients, m);

    int includeCurrent = 0, excludeCurrent
= 0;
    if (maxIngredientsWithThisBag >= k) {
        includeCurrent = maxIngredients(idx
- 1, m - maxIngredientsWithThisBag, k -
ingredients, bags, dp) +
maxIngredientsWithThisBag;
    }
    excludeCurrent = maxIngredients(idx -
1, m, k, bags, dp);
```

```
    return dp[idx][m] = max(includeCurrent,
excludeCurrent);
}

int main() {
    int n, m, k;
    cin >> n >> m >> k;

    vector<pair<int, int>> bags(n);
    for (int i = 0; i < n; ++i) {
        int ingredients, price;
        cin >> ingredients >> price;
        bags[i] = {ingredients, price};
    }

    vector<vector<int>> dp(n, vector<int>(m
+ 1, -1));

    int result = maxIngredients(n - 1, m,
k, bags, dp) + 1;

    if (result >= k) {
        cout << "YES " << result << endl;
```

```cpp
    } else {
        cout << "NO" << endl;
    }


    return 0;
}
```

Output:



```
int main() {
    vector<vector<int>> dp(n, vector<int>(m + 1, -1));

    int result = maxIngredients(n - 1, m, k, bags, dp
```

```
PS D:\Spring 241\CSE 2218\Assignment 02>
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ max
NumberOfIngredient.cpp -o maxNumberOfIngredient } ; if ($?) { .\maxNumberOfIngredient }
3 6 1
6 4
5 3
1 1
YES 7
PS D:\Spring 241\CSE 2218\Assignment 02> cd "d:\Spring 241\CSE 2218\Assignment 02\" ; if ($?) { g++ max
NumberOfIngredient.cpp -o maxNumberOfIngredient } ; if ($?) { .\maxNumberOfIngredient }
3 10 12
8 10
7 11
11 12
NO
PS D:\Spring 241\CSE 2218\Assignment 02>
```