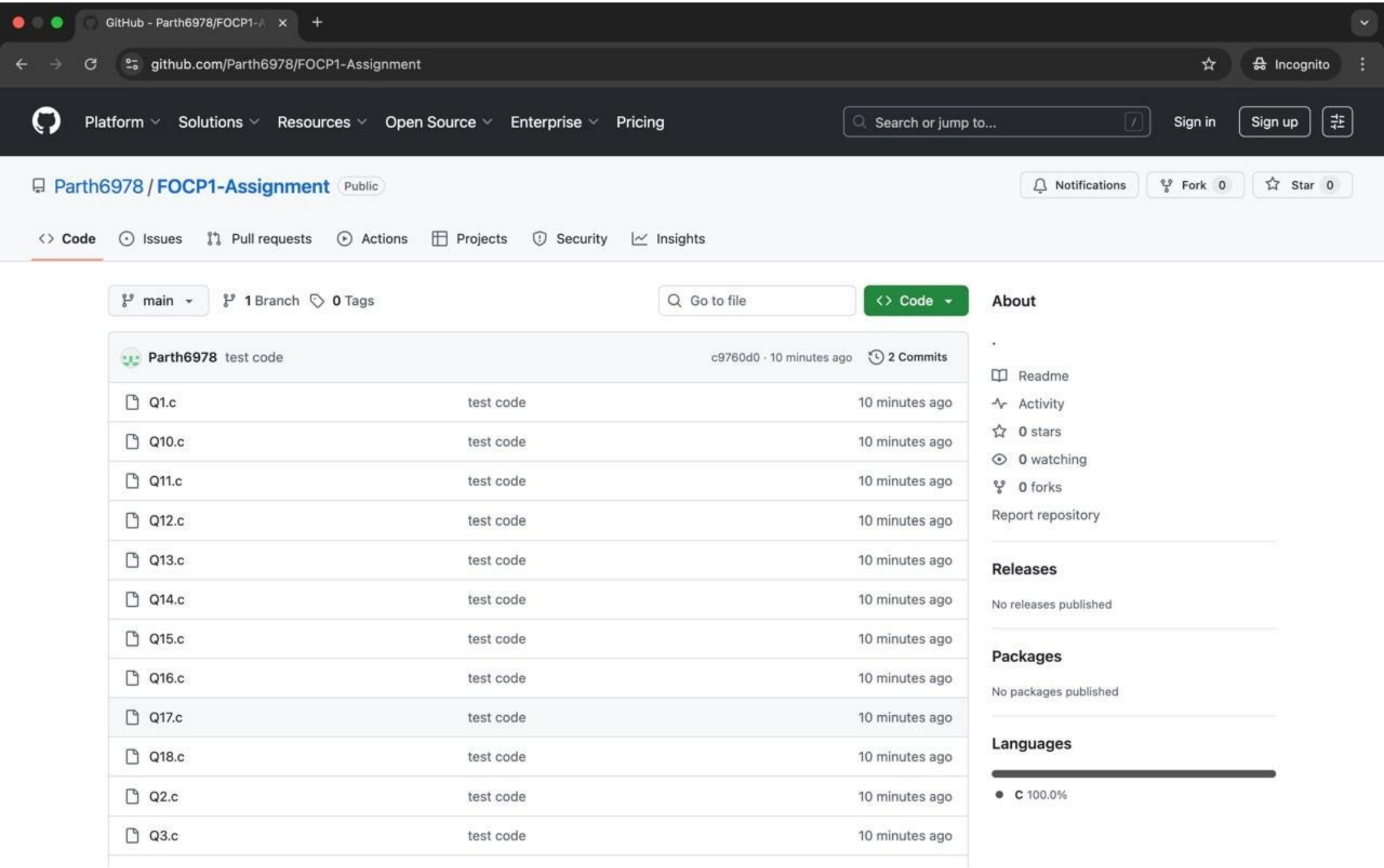

GitHub - Parth6978/FOCP1-A
github.com/Parth6978/FOCP1-Assignment
Incognito
Platform
Solutions
Resources
Open Source
Enterprise
Pricing
Search or jump to...
Sign in
Sign up
Parth6978 / FOCP1-Assignment
Public
Notifications
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Security
Insights
main
1 Branch
0 Tags
Go to file
Code
Parth6978 test code
c9760d0 · 10 minutes ago
2 Commits
Q1.c test code 10 minutes ago
Q10.c test code 10 minutes ago
Q11.c test code 10 minutes ago
Q12.c test code 10 minutes ago
Q13.c test code 10 minutes ago
Q14.c test code 10 minutes ago
Q15.c test code 10 minutes ago
Q16.c test code 10 minutes ago
Q17.c test code 10 minutes ago
Q18.c test code 10 minutes ago
Q2.c test code 10 minutes ago
Q3.c test code 10 minutes ago
About
Readme
Activity
0 stars
0 watching
0 forks
Report repository
Releases
No releases published
Packages
No packages published
Languages
C 100.0%

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

int main() {
    char word[21];
    scanf("%s", word);

    int z = 0, o = 0;

    // Count Zs and Os
    for(int i = 0; word[i] != '\0'; i++) {
        if(word[i] == 'z') z++;
        else if(word[i] == 'o') o++;
    }

    // Condition check
    if(2 * z == o) {
        printf("Yes");
    } else {
        printf("No");
    }

    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int L, N, W, H;
    scanf("%d", &L);
    scanf("%d", &N);

    while (N--) {
        scanf("%d %d", &W, &H);

        if (W < L || H < L) {
            printf("UPLOAD ANOTHER\n");
        }
        else if (W == H) {
            printf("ACCEPTED\n");
        }
        else {
            printf("CROP IT\n");
        }
    }
    return 0;
}
```

C Auto

```c
int singleNumber(int* nums, int numsSize) {
    int result = 0;
    for (int i = 0; i < numsSize; i++) {
        result ^= nums[i];
    }
    return result;
}
```

Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =

[2,2,1]

Output

1

Expected

1

Code

C Auto

```c
int searchInsert(int* nums, int numsSize, int target) {
    int left = 0, right = numsSize - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target)
            return mid;
        else if (nums[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return left;
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =

[1,3,5,6]

target =

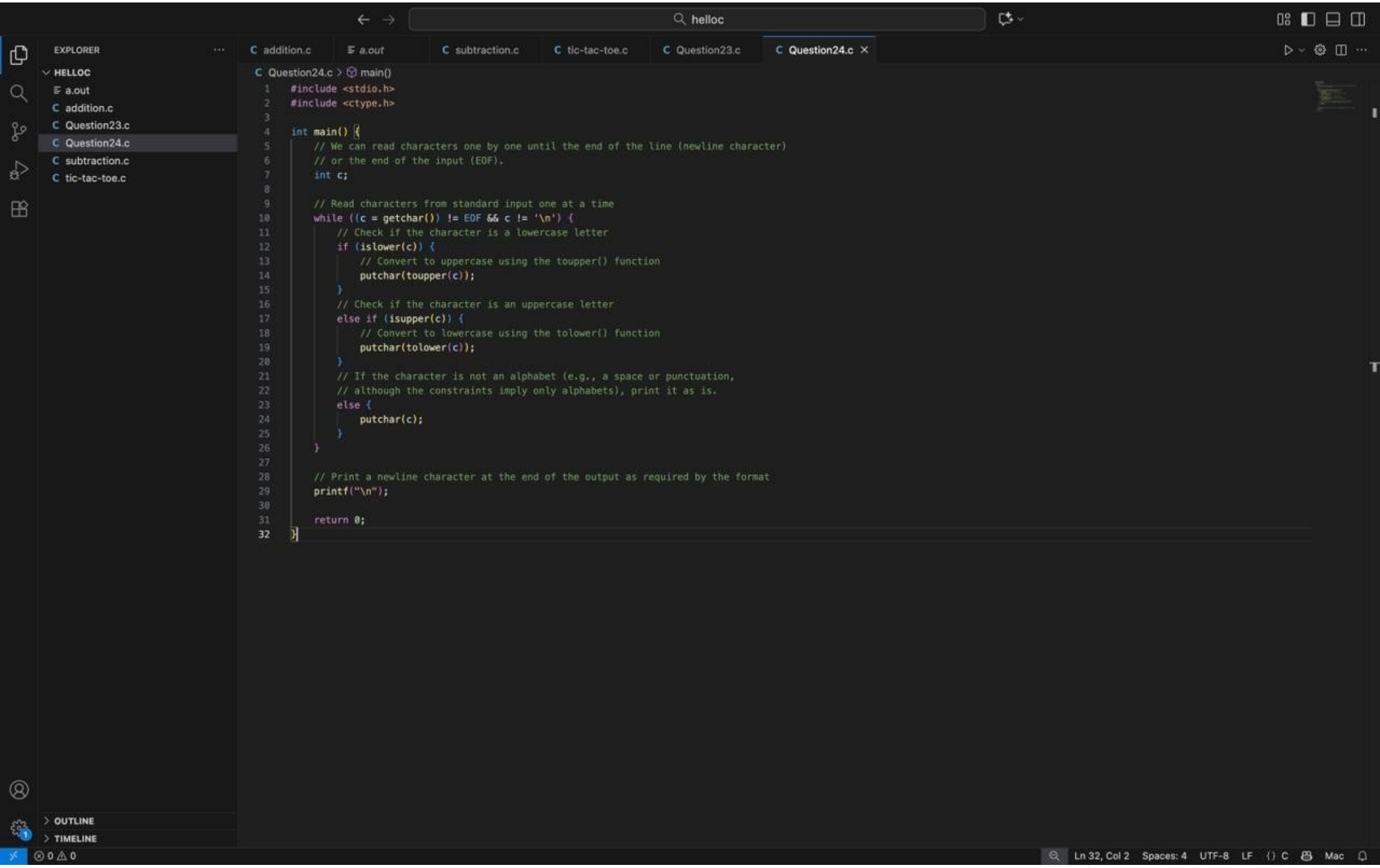
5

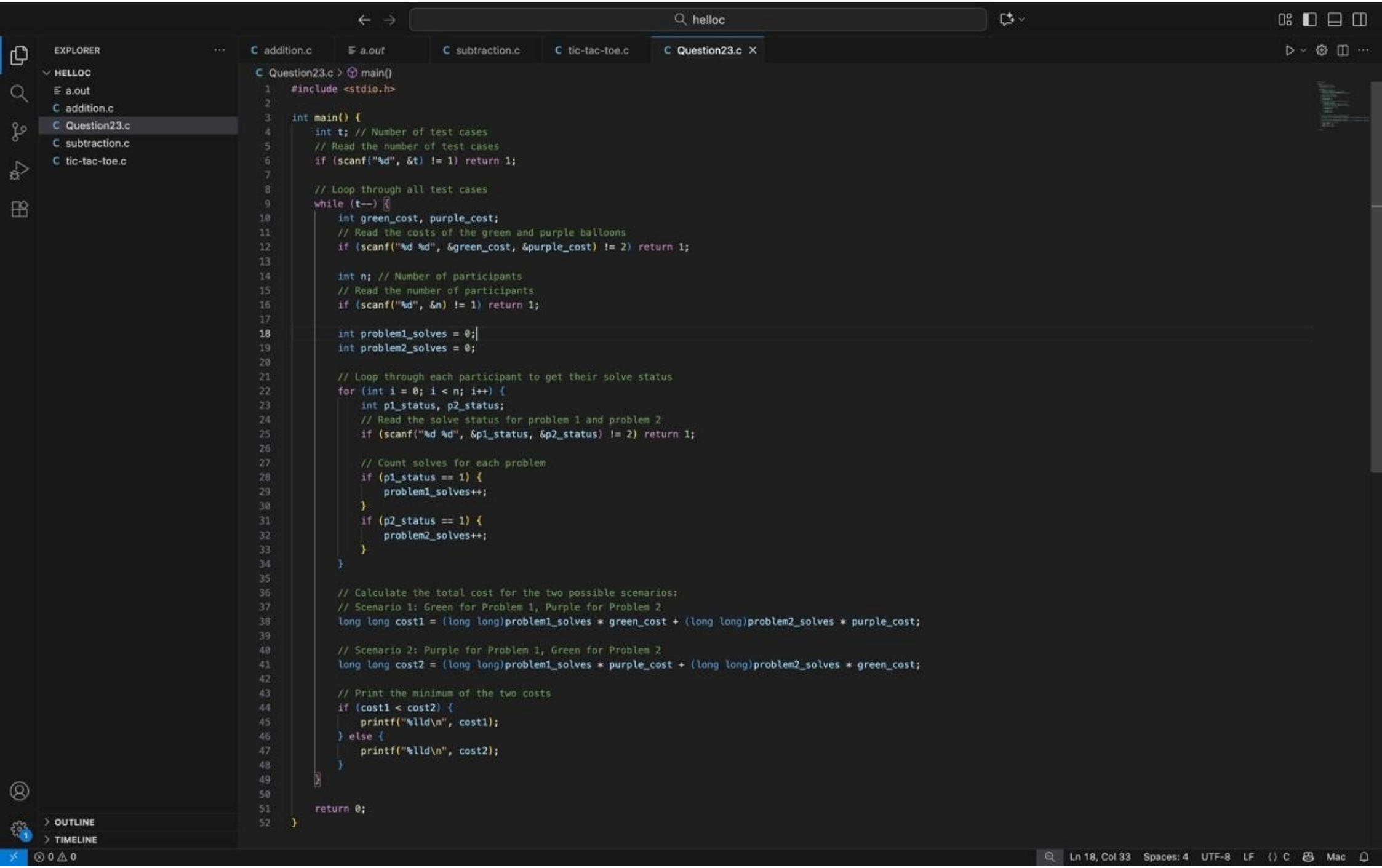
Output

2

Expected

2


helloc
EXPLORER
HELLOC
a.out
addition.c
Question23.c
Question24.c
subtraction.c
tic-tac-toe.c
addition.c
a.out
subtraction.c
tic-tac-toe.c
Question23.c
Question24.c
Question24.c > main()
#include <stdio.h>
#include <ctype.h>
int main() {
// We can read characters one by one until the end of the line (newline character)
// or the end of the input (EOF).
int c;
// Read characters from standard input one at a time
while ((c = getchar()) != EOF && c != '\n') {
// Check if the character is a lowercase letter
if (islower(c)) {
// Convert to uppercase using the toupper() function
putchar(toupper(c));
}
// Check if the character is an uppercase letter
else if (isupper(c)) {
// Convert to lowercase using the tolower() function
putchar(tolower(c));
}
// If the character is not an alphabet (e.g., a space or punctuation,
// although the constraints imply only alphabets), print it as is.
else {
putchar(c);
}
}
// Print a newline character at the end of the output as required by the format
printf("\n");
return 0;
}
OUTLINE
TIMELINE
0 0
Ln 32, Col 2
Spaces: 4
UTF-8
LF
C
Mac


helloc
EXPLORER
HELLOC
a.out
addition.c
Question23.c
subtraction.c
tic-tac-toe.c
addition.c
a.out
subtraction.c
tic-tac-toe.c
Question23.c
Question23.c > main()
#include <stdio.h>
int main() {
int t; // Number of test cases
// Read the number of test cases
if (scanf("%d", &t) != 1) return 1;
// Loop through all test cases
while (t--) {
int green_cost, purple_cost;
// Read the costs of the green and purple balloons
if (scanf("%d %d", &green_cost, &purple_cost) != 2) return 1;
int n; // Number of participants
// Read the number of participants
if (scanf("%d", &n) != 1) return 1;
int problem1_solves = 0;
int problem2_solves = 0;
// Loop through each participant to get their solve status
for (int i = 0; i < n; i++) {
int p1_status, p2_status;
// Read the solve status for problem 1 and problem 2
if (scanf("%d %d", &p1_status, &p2_status) != 2) return 1;
// Count solves for each problem
if (p1_status == 1) {
problem1_solves++;
}
if (p2_status == 1) {
problem2_solves++;
}
}
// Calculate the total cost for the two possible scenarios:
// Scenario 1: Green for Problem 1, Purple for Problem 2
long long cost1 = (long long)problem1_solves * green_cost + (long long)problem2_solves * purple_cost;
// Scenario 2: Purple for Problem 1, Green for Problem 2
long long cost2 = (long long)problem1_solves * purple_cost + (long long)problem2_solves * green_cost;
// Print the minimum of the two costs
if (cost1 < cost2) {
printf("%lld\n", cost1);
} else {
printf("%lld\n", cost2);
}
}
return 0;
}
OUTLINE
TIMELINE
Ln 18, Col 33
Spaces: 4
UTF-8
LF
C
Mac