**Campaign Project**

The Campaign project enables a user to find the correct campaign based on his provided choice of campaign segments.

The program takes a single argument, a file containing a list of campaigns. Once the program has loaded the data file, it will start accepting input from users.

**Step by step running of the program:**

Step 1:

The program first accepts and loads the campaign list file for reading.

The program accepts single file as an argument from the user.

Step 2:

The program first displays a map of campaign all the segments and its associates campaigns.

It then accepts user input i.e. segments.

The user should enter the input segments in the form of numbers. For eg. 3 4 10.

Note – No commas (,) or a delimiter other than space should be used.

Step 3:

Once the user enters his input, his input is split into string vector based on the provided delimiter.

The program will then return the most appropriate campaign for the provided user segments.

The program will keep accepting input from the user until the user exits by pressing ‘n’.

**Approach used:**

**Case 1.**

When user enters campaign segments, his input is split into string vector.

The program associates every segment to its respective campaign and maintains the count of each campaign. The campaign with highest count is returned to the user.

For eg. 3 4 5 10 2 200

User input Associated campaign Count

3 4 10 2 campaign\_a count - 4

3 campaign\_b count - 1

3 4 200 campaign\_c count – 3

Highest count campaign - campaign\_a

**Case 2.**

In order to deal with the issue of the campaign starvation, I have used an approach to maintain the count of the campaign and return the campaign with least count to the user.

For eg. The segment 3 is present in all 3 campaigns. We don’t want only one single campaign to be associated with it and starve two other campaigns. So, I maintain the count of campaigns every time it’s associated segment is provided by the user and return the campaign with least count.

User input Associated Campaigns Count

3 campaign\_a 0

campaign\_b 0

Campaign\_c 0

Output – Since the count of all the campaigns is same, a random campaign out of the three is returned. Let’s assume, it returns campaign\_c.

User input Associated Campaigns Count

3 campaign\_a 0

campaign\_b 0

Campaign\_c 1

Now, the user input again is 3. So, the program will return either campaign\_a or campaign\_b since count(campaign\_a) = count(campaign\_b).

**Case 3**.

When user passes segments which are not associated with any campaign, the program returns the output as: Campaign not found.

**Technologies used:**

Programming language – C++14

Compiler – Cygwin GCC5.0

IDE – Clion 2018.2

**Steps to run the program:**

Step 1:

Enter name (or path) of the text file as an argument to run the program.

Note – The program accepts both absolute and relative path

Step 2:

Enter inputs with space between them.

For eg. 3 4 10

Step 3:

Once you are done testing all the input you can exit the program.

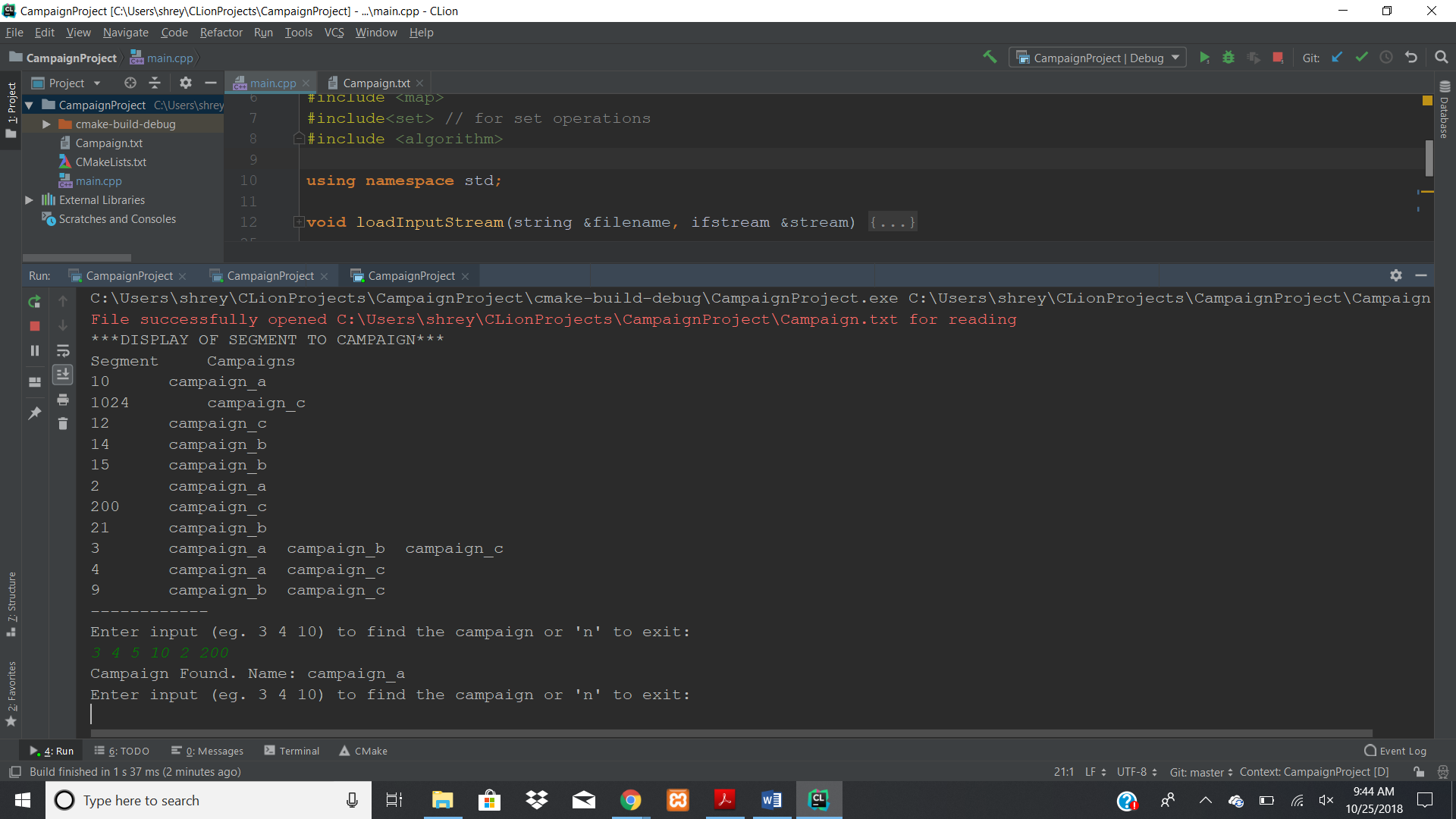
Press n to exit.

**Successful running of the program:**

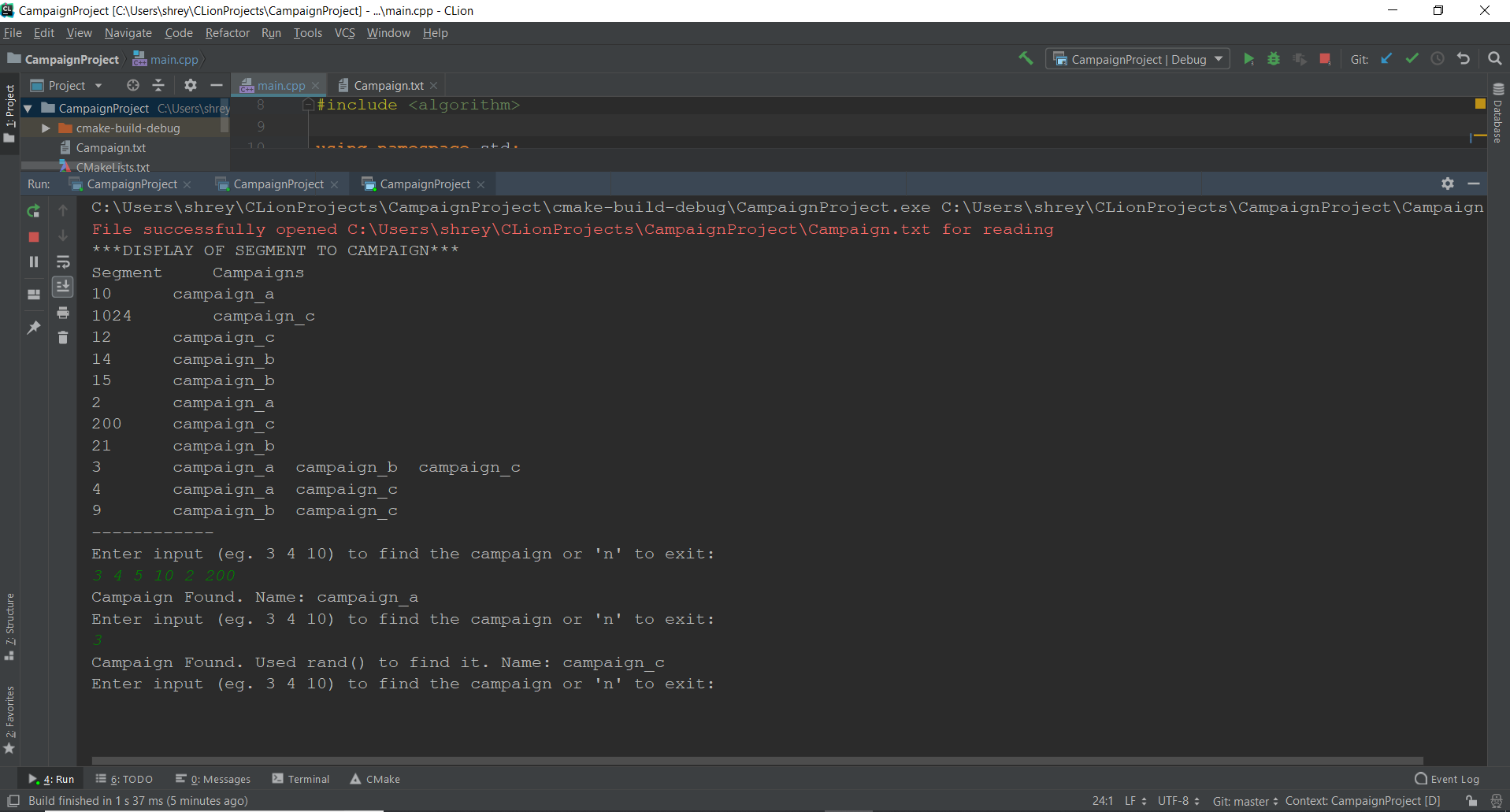
I’ve taken the same inputs as provided in the problem statement pdf file.

**Input 1**-> 3 4 5 10 2 200

Expected output – campaign\_a

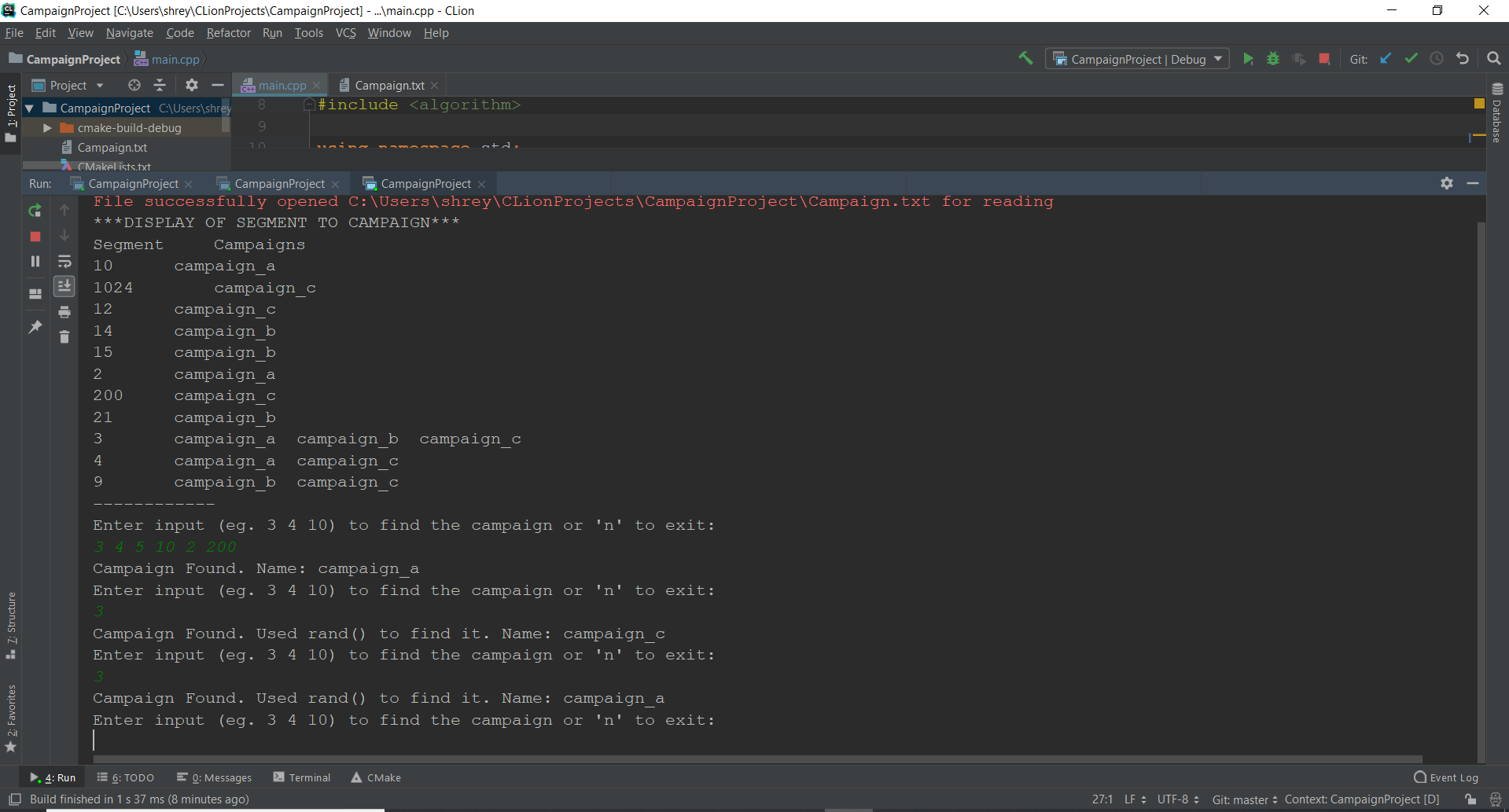


**Input 2**-> 3



Random campaign, campaign\_c is selected as the count of all the campaigns is same.

**Input 3** -> 3



Output – campaign\_a

Since the count after the second input 3 was:

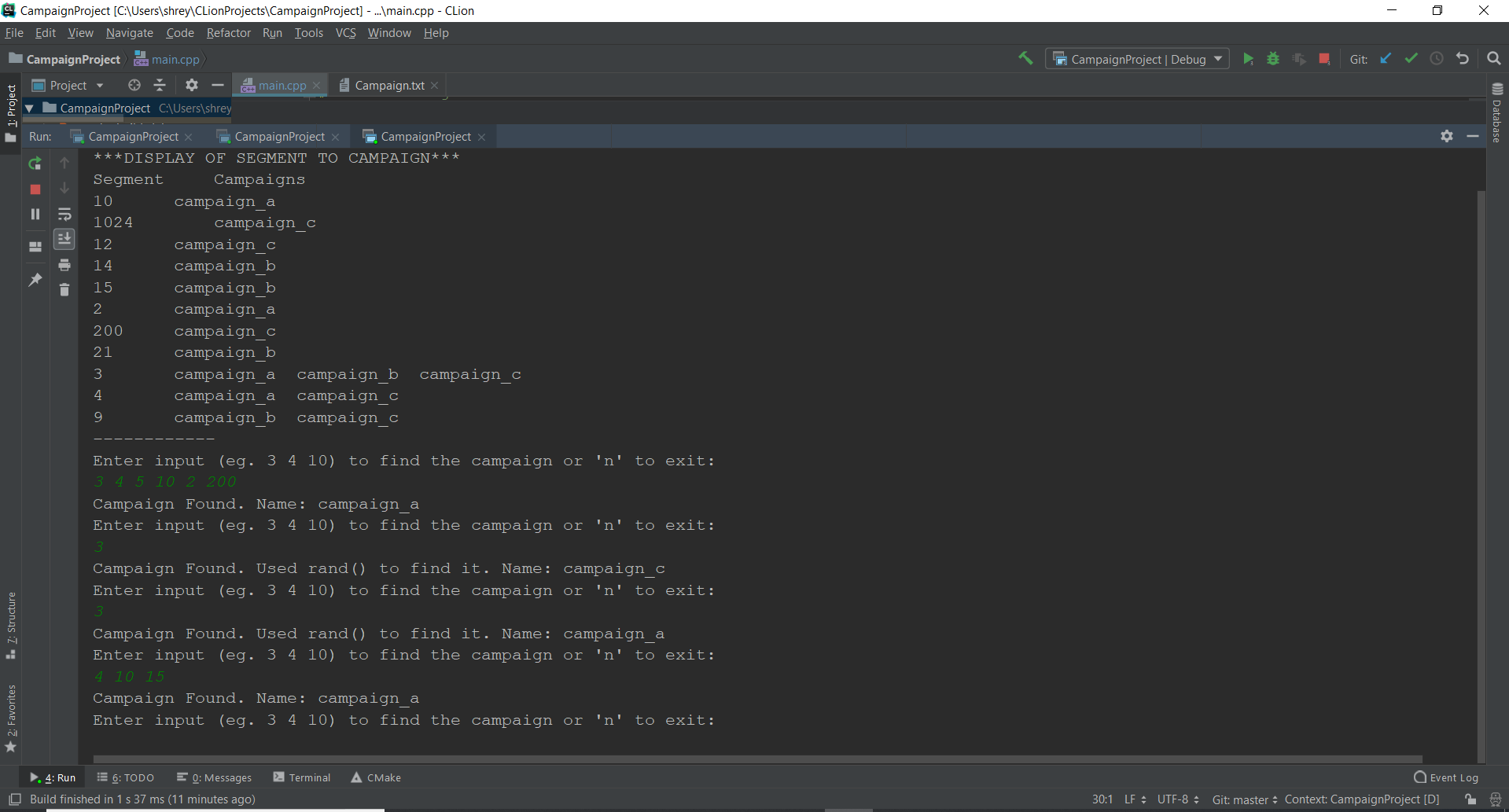
Campaign\_a -> 0

Campaign\_b -> 0

Campaign\_c -> 1

**Input 4** -> 4 10 15

Expected output – campaign\_a



**Input 5** -> 1024 15 200 21 9 14 15

Expected output – campaign\_b

In this input following is the breakdown of the campaigns and its segments

Segments Associated campaigns

1024 campaign\_c

15 campaign\_b

200 campaign\_c

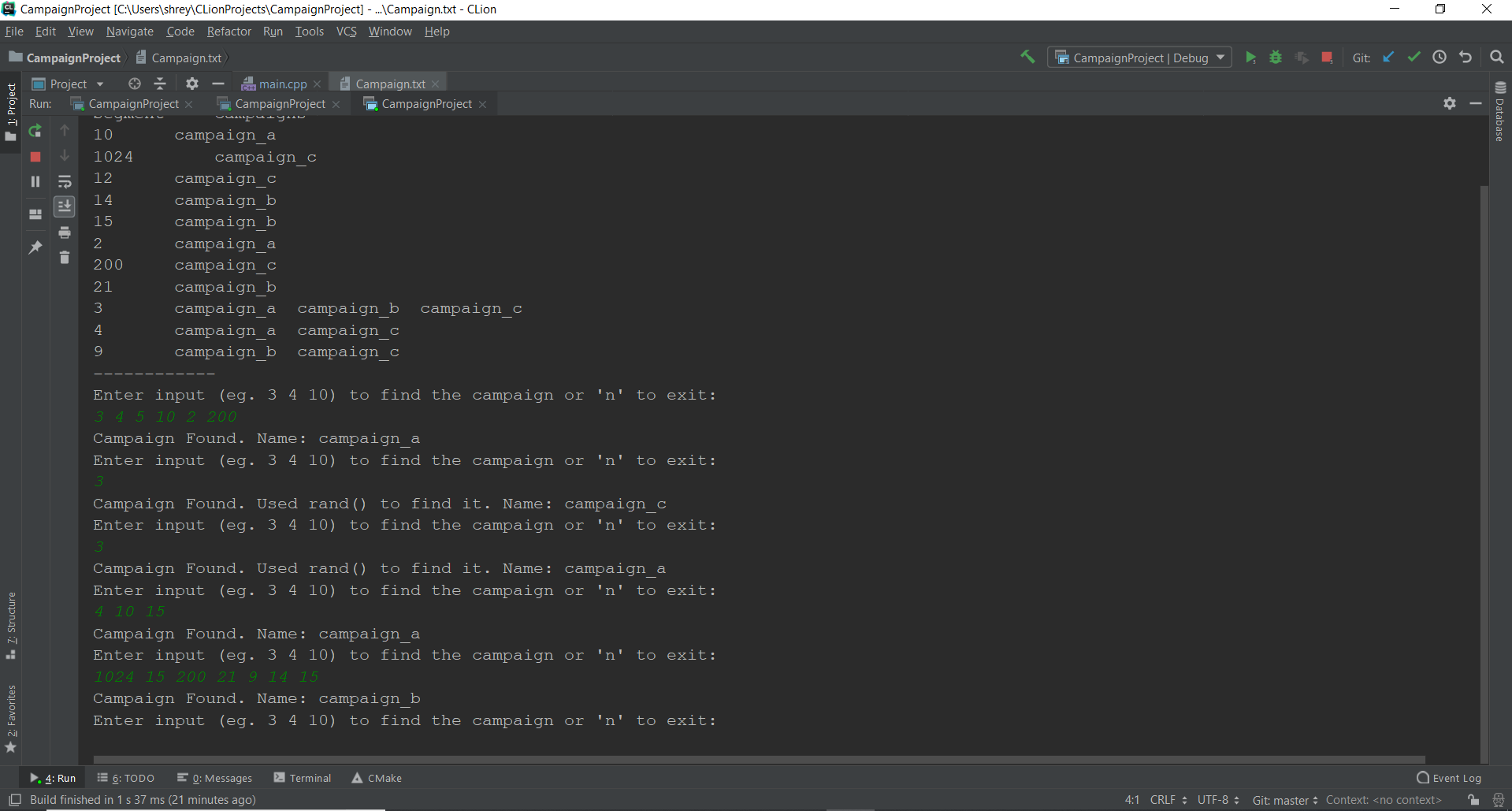
21 campaign\_b

9 campaign\_b campaign\_c

14 campaign\_b

15(duplicate) campaign\_b

Now, if we ignore the duplicate value 15 and the segment 9 is present is both campaign\_b and campaign\_c, if the program chooses campaign\_c, then count(campaign\_b) = count(campaign\_c). So, we select a random campaign\_b

****

**Input 6** -> 9000 29833 65000

Expected output -> Campaign Not Found

