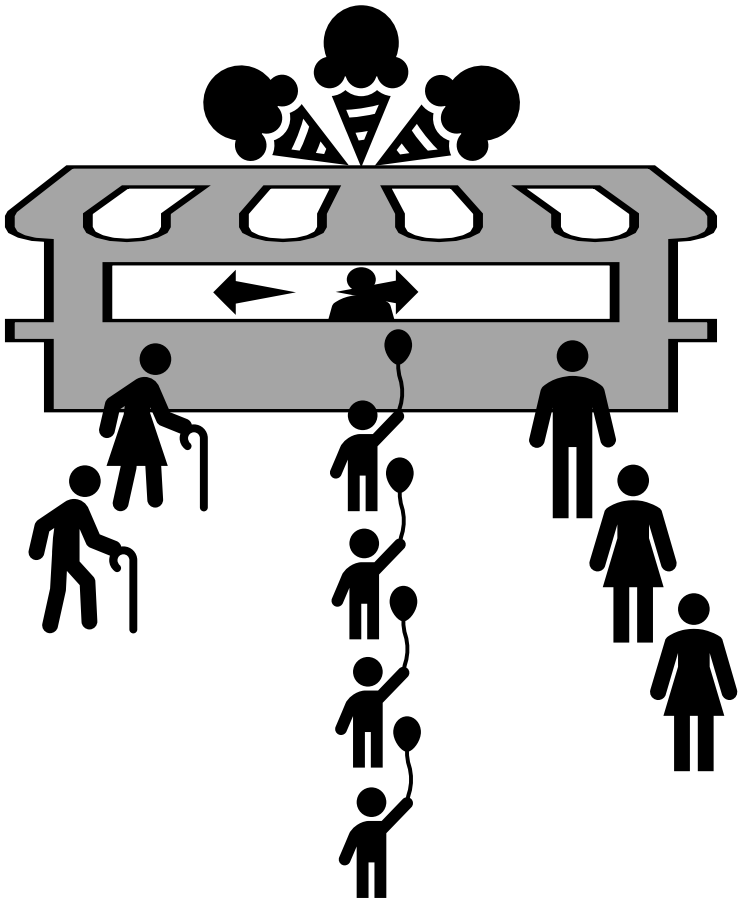
[redacted]

Submission Due Date: 06.12.2020 – 23:59

Development Environment: Standard C



INTRODUCTION - ICE CREAM SHOP

In an ice cream shop, normally, 3 employees work and each of them serves a different age group; elders, children and adults. Besides, they serve these different age groups from different payment points because each of these groups has different discount rates which are only valid at a specified one of these payment points.

However, 2 of the employees are unable to work today because of their health problems. Therefore, the remaining employee should also take place of the others and serve all the different age groups at their designated payment points. While doing this, he is repeatedly commanded by his boss to serve a specific number of customers from the beginning of either elders', children's or adults' queue. The employee continues to work by repeating this process until his shift is over while new customers arrive and queue up in their appropriate age group queues. At the end of the employee's shift, he prints a daily report of served customers.

EXPERIMENT TASKS

In this experiment, you are expected to develop a program that deals with services and report printing of the ice cream shop. Your program should read a text-based input file (named as "input.txt") whose lines correspond to commands that should be executed.

Commands

You may assume that each line will have a single command and that all command names and command parameters will be tab-separated.

- **NewCustomer:** Adds a customer to the specified age group queue. The format of this command is:

  NewCustomer <customer_age_group> <customer_name>

  <customer_age_group> indicates the age group of the new customer and can be one these values: "E" for elders, "C" for children and "A" for adults. <customer_name> indicates the name of the customer and can be a string having 10 alphabetic characters at most.

  Examples:

NewCustomer	E	Ramiz
NewCustomer	C	Elif
NewCustomer	A	Hakan

  This command has no output, so do not print anything to the screen or do not write anything to the output file.

- **ServeCustomers:** Serves a specified number of customers from the beginning of the specified age group queue. The format of this command is:

  ServeCustomers <customer_age_group> <number_of_customers>

  <customer_age_group> indicates the age group of the customers to serve and can be one these values: "E" for elders, "C" for children and "A" for adults. <number_of_customers> indicates the number of the customers to serve and can be a positive integer.

  Examples:

ServeCustomers	E	1
ServeCustomers	C	3
ServeCustomers	A	2

  This command has no output, so do not print anything to the screen or do not write anything to the output file. Nevertheless, while executing this command, you should record the information of each served customer, to be able to print them into the daily report. You can simply record age group and name of each customer served. However, the specified age group queue may be or may become empty while executing this command. In such a case, for each unsuccessful serve attempt, you can simply record the specified age group by the command and "*****" instead of the customer name.

Daily Report

When your program finished executing all of the commands given in the input file, it should print the daily report of served customers to a text-based output file (named as "output.txt") and then it should terminate. This daily report should include age groups ("E" for elders, "C" for children and "A" for adults) of the customers served and their names in a tab-separated format. The format of each daily report line is:

<customer_age_group> <customer_name>

It should start from the last served customer and continue towards the first one and each customer's information should be printed as a new line.

Example:

(*Assuming that the previously given sample commands are executed in the order they are given.*)

```
A     *****
A     Hakan
C     *****
C     *****
C     Elif
E     Ramiz
```

SAMPLE INPUT & OUTPUT

A sample input and its corresponding output is given below:

Input

```
ServeCustomers  E  1
ServeCustomers  C  2
ServeCustomers  A  1
NewCustomer     E  Ramiz
NewCustomer     C  Elif
NewCustomer     A  Hakan
ServeCustomers  E  2
NewCustomer     C  Tuğba
NewCustomer     C  Akın
NewCustomer     E  Mahmut
NewCustomer     A  Nermin
ServeCustomers  C  3
NewCustomer     E  Hatice
ServeCustomers  A  1
NewCustomer     C  Işıl
NewCustomer     C  Selim
NewCustomer     E  Kamil
NewCustomer     E  Melahat
ServeCustomers  E  2
ServeCustomers  E  1
NewCustomer     C  Aslı
NewCustomer     C  Nihat
NewCustomer     C  Nisa
NewCustomer     C  Kerem
NewCustomer     C  Tuna
NewCustomer     C  Seher
NewCustomer     C  Cenk
NewCustomer     C  Tuna
ServeCustomers  C  1
NewCustomer     C  Filiz
ServeCustomers  C  10
ServeCustomers  C  1
ServeCustomers  A  1
```

Output

```
A     Nermin
C     *****
C     Filiz
C     Tuna
C     Cenk
C     Seher
C     Tuna
C     Kerem
C     Nisa
C     Nihat
C     Aslı
C     Selim
C     Işıl
E     Kamil
E     Hatice
E     Mahmut
A     Hakan
C     Akın
C     Tuğba
C     Elif
E     *****
E     Ramiz
A     *****
C     *****
C     *****
E     *****
```

RULES & RESTRICTIONS

- There will not be any intentional syntactic or semantic errors in the commands given in the input file. As can be seen from the sample input and output, commands are independent of each other, so any command can be given at any time.
- Given customer numbers to serve in ServeCustomers commands may not be enough to serve all queued customers. In such a case, after finishing the execution of all given commands, information of not served customers should not be printed in the daily report, which is the case for elder "Melahat" in the sample input and output.
- For each of the age group queues, there may be 10 customers at most, at any time. There may be 100 customers at most, in a day (sum of customer counts specified in all ServeCustomers commands may be 100). You can adjust the limits of your data structures considering these numbers.
- The number of customers in each age group queue may become 10, and then may decrease to zero and then may become 10 again, and so on. So, you should choose suitable data structures to adapt to this situation.
- You should use the given constant input and output file names ("input.txt" and "output.txt") to read inputs from and to write results to. You may assume that the input file is located in the same directory with your executable. You should also create your output file in the same directory with your executable.
- At the beginning of your code file, as comment lines, you should explain which data structures you use, for which purposes and how you use them, in 10 sentences at most.
- You should only use standard C libraries and you should not use any other 3rd party codes or libraries. If your code requires any other libraries than the standard ones, your submission will not be evaluated.
- You should implement yourself all the data structures and algorithms you need. Otherwise, your submission will not be evaluated.
- Your experiment will be compiled and executed using version 20.03 of Code::Blocks (with MinGW-W64 version 8.1.0). So, you should make it work on this configuration before submitting. Otherwise, your submission will not be evaluated.
- The experiment must be original, individual work. Duplicate or very similar experiments will be threated according to regulations. Do not discuss the problem with classmates and do not share answers, algorithms or source codes. Keep in mind that you will get points for every little thing you do.
- You can ask your questions about the experiment via sending your messages via LMS. Any other attempts will be discarded.
- Your submission may not be evaluated or you can get punishment points if you do not follow the rules defined above.

SUBMISSION

- You should submit only one .c code file that is named with your student ID (<student_ID>.c) and that file should include all of your code. So, do not separate your code into multiple code files. Otherwise, your submission will not be evaluated.
- You should upload your code file for the experiment under course page via LMS before submission due.
- Do not submit any file via e-mail. Submissions via e-mail will not be evaluated.
- No submission will be accepted after deadline.
- Save all your work until the experiment is graded.