Super Maria II: Another Castle

The Fungus Republic had been a peaceful place thanks to the brave deeds of the humble electrician, Maria, but this all changed when her arch-nemesis Wowser invaded. Wowser kidnapped Prince Nectarine (again) and hid him somewhere he thought Maria would never find.

Maria was undaunted, however, and using the signal from Nectarine's mushroom phone was able to narrow down his location to one of N castles. The castles are arranged in a line and are numbered from 0 to $N-1$ where castle 0 is the leftmost castle, castle 1 is second from the left, etc.

Maria is very hasty about finding the prince, so she has rented a carriage to be pulled by Shoyi the friendly dinosaur. Whenever Maria wants to get out of the carriage or change direction, she must pay Shoyi a tip.

Maria begins at castle 0, facing to the right. When Maria is at castle i, she can do one of three things.

- Move to any castle in the direction she is facing without spending any coins. That is, if she is facing left, she can move to any castle j at no cost providing that $j < i$. If she is facing right then she can move to any castle j at no cost providing that $j > i$.
- Pay Shoyi C_1 coins to turn the carriage around and face the opposite direction.
- Pay Shoyi C_2 coins to let Maria out so that she can open the door of castle i. If Maria chooses to do this, one of two things happen.
  - She finds Prince Nectarine, defeats Wowser, and returns the Fungus Republic to safety.
  - Shce finds Frog, who kindly tells her that the prince is in another castle and in which direction (left or right) she must go to find him.

Maria wants to minimise the total number of coins she spends in finding the prince. Maria whips out her trusty laptop and sends you an e-mail, asking you to write a program to help find the prince while minimising the total number of coins she pays Shoyi.

Input / Output

This task has no input or output files. Instead it must use functions from the header file "`carriage_lib.h`" to gain information. These provided functions are described in detail on the next page.

Library

Your program must interact with functions in "`carriage_lib.h`", as follows:

- Your program must begin by calling `carriage_nb_cells()`, `carriage_reverse_cost()`, and `carriage_open_cost()`
  - `carriage_nb_cells()` returns N, the number of castles which the prince may be in.
  - `carriage_reverse_cost()` returns C_1, the number of coins Maria must pay Shoyi to change direction.
  - `carriage_open_cost()` returns C_2, the number of coins Maria must pay Shoyi to stop so that she can open a castle door.
- Your program should then repeatedly call `carriage_move(int k)` and `carriage_open()` in order to find the castle which holds the prince.
  - `carriage_move(int k)` moves the carriage a distance of $|k|$ to the left or right depending on whether k is positive or negative. For instance calling this function with -5 as an argument will cause Maria's carriage to move 5 cells to the left. If Maria is not already facing in the direction she is made to move, she must pay Shoyi C_1 coins to change direction. If the argument given to this function causes Maria to move to a position which is negative or is not less than N, then your program will be terminated and will be awarded no marks.
  - `carriage_open()` causes Maria to pay Shoyi C_2 coins to stop the carriage so that she can open the castle door. If Maria is at the castle where Prince Nectarine has been hidden then your program will be terminated and you will be awarded marks according to the "Scoring" section on the next page. Otherwise, this function will return 1 if the prince is in a castle to the right of the current castle, or -1 if the prince is in a castle to the left of the current castle.

Library Prototypes

The signatures for these functions in C and C++ are as follows:

```
int carriage_nb_cells();
int carriage_reverse_cost();
int carriage_open_cost();
void carriage_move(int k);
int carriage_open();
```

Experimentation

In order to experiment with your code on your own machine, first download `carriage_lib.h` to the same folder as your code file and add `#include "carriage_lib.h"` to the top of your code.

The compiled executable will take four whitespace-separated integers from standard input on the first call to a provided function from the header file. The integers should represent, respectively, N, C_1, C_2 and p where the castle with index p (between 0 and $N-1$) contains Prince Nectarine.

The executable will print one line to standard output, either an error message due to invalid input or an invalid move, or an indication of success including the number of coins used.

Sample Input

```
10 3 4 7
```

Sample Session

Function Call	Explanation
`carriage_nb_cells()`	Returns 10, the number of castles.
`carriage_reverse_cost()`	Returns 3, the cost of changing direction.
`carriage_open_cost()`	Returns 4, the cost of opening a castle door.
`carriage_move(5)`	Moves to castle 5, at no cost.
`carriage_open()`	Returns 1, as the Prince is in castle 7, to the right of castle 5. This costs 4 coins.
`carriage_move(4)`	Moves to castle 9, at no cost.
`carriage_open()`	Returns -1, as the Prince is in castle 7, to the left of castle 9. This costs 4 coins.
`carriage_move(-3)`	Moves to castle 6. This costs 3 coins, as it involved a change in direction.
`carriage_open()`	Returns 1, as the Prince is in castle 7, to the right of castle 6. This costs 4 coins.
`carriage_move(1)`	Moves to castle 7. This costs 3 coins, as it involved a change in direction.
`carriage_open()`	Finds the prince! This costs 4 coins. The program outputs that the prince has been found with the use of 22 coins and exits.

Scoring & Constraints

- For all cases, N, C_1 and C_2 are integers such that $1 \leq N \leq 1,000,000$ and $0 \leq C_1, C_2 \leq 10,000$.
- For 30% of cases, $N \leq 1000$ and $C_1, C_2 \leq 1000$.
- For each test case, your score will be 100 if your program finds the Prince with the minimum number of coins. Otherwise, if your program successfully finds the Prince your score will be a value between 0 and 60, based on a linear function of the number of coins your program used.
- Your total score for the task will be the average of your scores for each individual test case.
- Note that the judges use a modified version of the library which will force your program to use as many coins as possible in each case.