Question 3: *String rewriting*

A *string rewriting rule* is a simple instruction for changing a string. It says that every occurrence of a given character should be changed into one or more other characters. In this question we will use five rules:

- `A` should be changed into `B`
- `B` should be changed into `AB`
- `C` should be changed into `CD`
- `D` should be changed into `DC`
- `E` should be changed into `EE`

In a single step, we apply these rules simultaneously and on every character in a string. For example, suppose we have the string `DECADE`, after one step we have the string `DCEECDBDCEE`, and after a second step we have the string `DCCDEEEECDDCABDCCDEEEE`.

3(a) [25 marks]

Write a program to determine the number of each type of character in part of a string after a given number of steps.

Your program should input two lines, the first containing a three letter string (where each letter will be `A`, `B`, `C`, `D` or `E`), and the second containing two integers s ($1 \le s \le 29$) followed by p ($1 \le p \le 2^{31}$). The first integer s indicating the number of rewriting steps which should take place and the second p indicating a position.

Sample run

```
DEC
2 10
0 0 3 3 4
```

You should output 5 numbers, the number of `A`s (then `B`s, `C`s, `D`s and `E`s) occurring in the first p (inclusive) positions of the string after s string rewriting steps.

The value p will never be larger than the number of characters in the string after s steps.

3(b) [2 marks]

How many letters will there be if you start from the string `C` and apply 8 steps? How about if you start from the string A?

3(c) [5 marks]

Given any starting position and number of steps (>1) is it ever possible to have three adjacent `C`s in the resulting string? Justify your answer.

3(d) [3 marks]

For some strings we can go backwards; i.e. we can find a previous string which leads to it after applying a step. With the rules given in this question, when we are able to move backwards there is always a *single* possible previous string.

For some sets of rules the situation might be ambiguous; when we try to move backwards there many be more than one possible previous string.

Find the smallest set of rules (both in terms of the number of rules and the number of characters used in those rules) which makes things ambiguous when moving backwards. Your rules must allow growth, so given any starting string it must be possible to apply enough steps to make the length of that string increase.

Total Marks: 100 End of BIO 2007 Round One paper