2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

Betting PROBLEM STATEMENT

In the game of poker, players have a chance to bet. N players numbered 0 through N-1 sit in a circle. Betting starts with player 0 and proceeds in the circle. That is, player 0 is followed by player 1, then 2, ... and player N-1 is followed by player 0. Initially, the current bet is 0 and each player has put in 0 chips. When it is a players turn, the player can "call" the bet, "raise" the bet by X chips, or "fold":

*If the player calls, then he or she puts in as many chips as necessary to match the current bet. So for example, if 15 is the current bet and the player has already put in a total of 5 previously during the round, the player must put in 10 more to call. If the current bet is 0, then the player does not have to put in chips to call.

*If the player folds, then he or she is out of the game and does not put in more chips. A player cannot fold when the current bet is 0. After folding, the player's turn is skipped for the remainder of the betting, and the player does not get chips back.

*If the player raises the bet by X chips, then he or she first puts in the amount of chips necessary to call, and then X extra chips. This increases the current bet by X, and thus increases the amount that other players need to put in to call. The total number of chips put in by the player is the new current bet.

A round of betting is over when:
1) everyone but one player has folded, or
2) everyone has had a chance to bet and everyone that has not folded has put in a total number of chips equal to the current bet, or
3) A special case of 2) is if each player has had a chance to bet and everyone chose to call. This would be represented as a String composed of only 'C's.

To demonstrate, take the following 3 player betting round:

```
Player Move      Chips added to pot       Current bet
1      call      0                        0
2      call      0                        0
3      raise 4   4                        4
1      call      4                        4
2      raise 5   9                        9
3      call      5                        9
1      fold      0                        9
```

At this point player 2 and 3 are both in, and since both of them have put in a total of 9 chips (the current bet), the round is over. There are a total of 22 chips in the pot (the "pot" is the term for the total number of chips put in by all players combined).

Given a String representing a betting pattern, return the number of chips in the pot (defined above). The pot starts at 0, and the current bet starts at 0. The String will represent a complete round of betting and will end with the round ending, as defined above.

The String will contain the numbers [1-9] inclusive, and the following uppercase characters:
C - call

R# - raise # of chips (single digit raise, from 1 to 9 chips)
F - fold

The above example would be represented by the String (quotes for clarity):
"CCR4CR5CF"

You do not know how many players are in the game, but are guaranteed that the String will work for one (and only one) finite number of players greater than 1. That is, the String will represent a complete round of betting and will contain no extra characters beyond those needed to complete the round.

DEFINITION
Class name: Betting
Method name: potSize
Parameters: String
Returns: int
The method signature is:
int potSize(String sequence)
Be sure your method is public.

TopCoder will ensure the following:
*sequence will contain between 2 and 50 characters, inclusive.
*sequence will only contain the numbers [1-9], inclusive, and the capital characters 'C', 'R', and 'F'.
*only the character 'R' may (and always will) be followed by a number, and even then only by a single digit number.
*sequence will represent a valid and complete betting pattern (so that the betting round ends, as defined above) for a unique finite number of players greater than 1.
*a player will not fold if no bets have been placed (i.e. the current bet is 0).

Notes:
-The order of the players is maintained. That is, player i follows player i-1, unless he has folded.
-If a player has folded, then no more characters in the String refer to that player (see last example).
-The players are sitting in a circle, so player 1 follows the last player, unless he has folded.
-If a player has folded, the bet keeps passing to the next player until it reaches a player who has not yet folded. There will never be a situation where every player has folded. The last example illustrates this point.

EXAMPLES
(quotes added for clarity)

1)
"CCR4CR5CF"
There are 3 players (there is no other option).

```
Player Move      Chips added to pot      Current bet
1      call      0                       0
2      call      0                       0
3      raise 4   4                       4
1      call      4                       4
2      raise 5   9                       9
3      call      5                       9
1      fold      0                       9
```

return = 22

2)

```
"CC"
return = 0

3)
"R5R5R5C"
There are 2 players.
Player Move      Chips added to pot      Current bet
1      raise 5   5                       5
2      raise 5   10                      10
1      raise 5   10                      15
2      call      5                       15
Note that you could assume 4 players.  If you had, then it would have been:
Player Move      Chips added to pot      Current bet
1      raise 5   5                       5
2      raise 5   10                      10
3      raise 5   15                      15
4      call      15                      15
But this is not a complete betting round, since players 1 and 2 never called
player 3's raise, and thus have not yet matched the current bet.
return = 30

4)
"R5R5R5F"
There are 2 players.
return = 25

5)
"CCR5CFC"
There are 4 players.
return = 15

6)
"R2FFR2CCCCCCR9FFFFR1CCR9CCCFFR2FFFFCR2CR8FF"
There are 15 players.
return = 180

7)
"CCR2CR2R4FCCR2CCFR2CCR8FCC"
There must be 6 players:
Player Move      Chips added to pot      Current bet
1      call      0                       0
2      call      0                       0
3      raise 2   2                       2
4      call      2                       2
5      raise 2   4                       4
6      raise 4   8                       8
1      fold      0                       8
2      call      8                       8
3      call      6                       8
4      raise 2   8                       10
5      call      6                       10
6      call      2                       10
Since player 1 folded, the play then passes to player 2.
2      fold      0                       10
3      raise 2   4                       12
4      call      2                       12
5      call      2                       12
6      raise 8   10                      20
Since player 1 and player 2 both folded, the play then passes to player 3.
```

```
3      fold     0                       20
4      call     8                       20
5      call     8                       20
```

```
At this point, the remaining players (4, 5, and 6) have all matched the current
bet of 20, and the round of betting is over.
return = 80
```