

String Questions

This question involves finding double-letter pairs in a string. A double-letter pair means the same letter twice in a row. For example, the string "aabccde" has two double-letter pairs: "aa" and "cc".

Part (a)

Write function *DoublePosition*, as stated below. *DoublePosition* should find the first double-letter pair in *S* and should return the position of the first two letters. If *S* does not contain any double-letter pairs, *DoublePosition* should return -1.

For example:

<u>String <i>S</i></u>	<u>Result of the call <i>DoublePosition</i>(<i>S</i>)</u>
"happy"	2
"aardvark"	0
"aabcc"	0
"abc"	-1

Complete function *DoublePosition* below. Assume that *DoublePosition* is called only with values that satisfy its precondition.

```
int DoublePosition(const apstring & S)
// pre: S contains only lowercase letters
// post: returns the position of the first letter of the first double-letter
//       pairs in S; if there are no double-letter pairs in S, returns -1
```

Part (b)

Write function *MoveLeft*, as stated below. *MoveLeft* should move the characters in string *S* in positions *k* through *S.length() - 1* one place to the left, overwriting the character in position *k - 1*. (Assume that, as specified by *MoveLeft*'s precondition the value of *k*'s is in the range 1 to *S.length() - 1*.) Note that this makes *S* one character shorter than it was originally.

For example:

<u>Initial value of String <i>S</i></u>	<u>Function Call</u>	<u>Value of <i>S</i> after the call</u>
"abcde"	<i>MoveLeft</i> (<i>S</i> , 1)	"bcde"
"abcde"	<i>MoveLeft</i> (<i>S</i> , 2)	"acde"
"abcde"	<i>MoveLeft</i> (<i>S</i> , 3)	"abde"
"abcde"	<i>MoveLeft</i> (<i>S</i> , 4)	"abce"

In writing *MoveLeft*, you may make use of the *substr* member function of the *apstring* class, specified as follows.

```
apstring substr(int pos, int len) const
// returns the substring of len characters, starting at position pos
// pre: this string contains the characters  $c_0 c_1 \dots c_n$ 
//      0  $\leq$  pos  $\leq$  pos + len - 1  $<$  n
// post: returns a string containing the characters
//        $c_{pos} c_{pos+1} \dots c_{pos+len-1}$ 
```

Complete function *MoveLeft* below. Assume that *MoveLeft* is called only with values that satisfy its precondition.

```
void MoveLeft(apstring & S, int k)
// pre: 0  $\leq$  k  $\leq$  S.length()
//       $c_0 c_1 \dots c_n$ 
// post: S contains the characters  $c_0 c_1 \dots c_{k-2} c_k c_{k+1} \dots c_n$ 
```

Part (c)

Write function *RemoveDoubles*, as stated below. *RemoveDoubles* should find all double-letter pairs in *S* and convert them to letters by moving all of the characters to the right of the double-letter pair one place to the left (making *S* one character shorter each time).

For example:

<u>Initial value of S</u>	<u>Value of S after the call RemoveDoubles(S)</u>
"happy"	"hapy"
"aabcc"	"abc"
"aaabc"	"abc"

In writing *RemoveDoubles*, you may include calls to functions *DoublePosition* and *MoveLeft*, as specified above in parts (a) and (b). Assume that *DoublePosition* and *MoveLeft* work as specified, regardless of what you wrote for parts (a) and (b).

Complete function *RemoveDoubles* below. Assume that it is called only with values that satisfy its precondition.

```
void RemoveDoubles(apstring & S)
// pre: S contains only lowercase letters
```