CS29003 ALGORITHMS LABORATORY

LAB TEST | ODD PC | 13-November-2014

Read an integer $n < 100$ and a string S of length n from the user. Write two functions *norepdecomp*() and *longestpalsubstr*() to solve the following parts.

Part I

Decompose S as a concatenation of strings $S_1, S_2, \ldots, S_k$ such that no S_i contains the same symbol in consecutive positions, and k is as small as possible. For example, for the input string *bbbcaacacabacccaccbb*, your function should report $k = 8$ and the decomposition *b*, *b*, *bca*, *acacabac*, *c*, *cac*, *cb*, *b*.

Part II

Find longest palindromic substrings of S. If there are multiple palindromic substrings of the same maximum length, report all of them. For example, for the input S = *bbbcaacacabacccaccbb*, you should report the four substrings *acaca*, *cabac*, *accca*, and *ccacc*. Your program must run in $O(n^2)$ time and use only $O(n)$ additional space. The obvious $O(n^3)$-time algorithm of checking whether each of the $\Theta(n^2)$ substrings of S is palindromic will deserve no credit.

Hint: You need to efficiently compute which of the substrings of S are palindromic.

Sample Output

```
n = 50
S = bbbabbcbccbaaaaaccbcababccabbcabcccaacbacabcbcbabc
Part I: b b bab bcbc cba a a a ac cbcababc cab bcabc c ca acbacabcbcbabc
Part II: Length = 7. Substrings: cbacabc abcbcba
```

CS29003 ALGORITHMS LABORATORY

LAB TEST | EVEN PC | 13-November-2014

Read an integer $n < 100$ and a string S of length n from the user. Write two functions *norepsubseq*() and *paldecomp*() to solve the following parts.

Part I

Find a longest subsequence T of S such that T does not contain the same symbol in consecutive positions. For example, for the input string *bcbcabacbaccbcbaabbb*, T should be *bcbcabacbacbcbab*.

Part II

Decompose S as a concatenation of strings $S_1, S_2, \ldots, S_k$ such that each S_i is a palindrome, and k is as small as possible. For example, for the input S = *bcbcabacbaccbcbaabbb*, your function would report $k = 8$ and the possible decomposition *b*, *c*, *bcabacb*, *a*, *cc*, *bcb*, *aa*, *bbb*. There may be multiple decompositions having the same optimal k. Reporting any of the optimal decompositions suffices. Your function must run in $O(n^2)$ time and use only $O(n^2)$ additional space.

Hint: You need to efficiently compute which of the substrings of S are palindromic.

Sample Output

```
n = 50
S = cabacabaacbbcaaabaaaabccbcbcabcccacccbaabbbbbacbcca
Part I: cabacabacbcababcbcbcabcacbabacbca
Part II: 12 substrings: c abacaba acbbca aa baaaab c cbcbc abcccacccba abbbba cbc c a
```