ES 314 Advanced Programming, Modeling and Simulation Fall 2011

Mid-term # 2, take-home test, Due: Nov 22, 8 AM

Create a zip file containing your programs with names indicated in the problems and e-mail it to same996@gmail.com or attach each of your program as a matlab file (such as count.m, palindrome_game.m etc.)

Your code should follow the syntax of Matlab exactly and should take the parameters as specified in the problem statement. Add comments to your code as appropriate.

1) Write a program **count** that takes as input a string and computes the following: (a) length of the string, (b) the number of distinct letters, (c) the number of occurrences of each letter in the string, and (d) the number of occurrences of adjacent pairs of characters.

Example:

```
>> count('thisisastringstring')
Total number of letters =19
Number of distinct letters =8
count(t)=3
count(th)=1
count(h)=1
count(hi)=1
count(i)=4
count(is)=2
count(s)=4
count(si)=1
count(sa)=1
count(a)=1
count(as)=1
count(st)=2
count(tr)=2
count(r)=2
count(ri)=2
count(in)=2
```

```
count(n)=2
count(ng)=2
count(g)=2
count(gs)=1
```

Note: The input string can contain any non-blank character.

2) Start with any positive integer x (for the purpose of this problem, we can limit to 32-bit integers) and generate successive integers by adding the previous number to its reverse. It is known that eventually the sequence will reach a palindromic integer, i.e., an integer that is a reverse of itself.

Example: suppose the starting number is 2993. The reverse is 3992. Adding the two we get 6985. Repeating, we get 12881, 31702, 52415, 103840 and finally 152141 which is a palindrome.

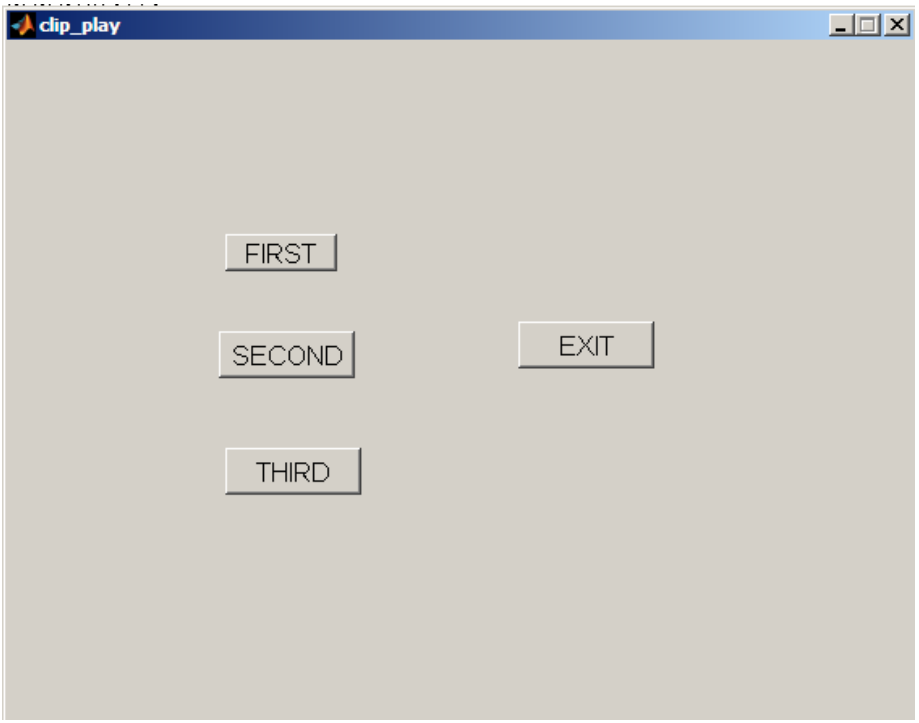
Another example:

```
>> palindrome_game(8712)

ans =

        8712    10890    20691    40293    79497
```

Write a function **`palindrome_game`** that takes as input an integer N, and produces as output the successive numbers generated by the above process ending with a palindrome.

3) Assume that there are three audio clips 'first.wav', 'second.wav' and 'third.wav' in the current directory. Write a script **`clip_play`** such that when it is run, it creates a GUI with five buttons labeled FIRST, SECOND, THIRD and EXIT. When one of the first three buttons is pressed, the corresponding audio clip should be played. When EXIT is pressed, the program is terminated.

Create a simple GUI like the one shown below:



4) A ***Latin square*** is an n by n matrix (for some n) that has exactly one occurrence of every number from 1 to n in every row, and in every column. Shown below is an example of a Latin square:

```
1   2   4   3
2   3   1   4
3   4   2   1
4   1   3   2
```

Two arrays A and B both of order n is said to form an ***orthogonal Latin Square pair*** if both A and B are Latin squares, and the pairs (a[i,j], b[i,j]) are all distinct. The following is an example of a 5 by 5 orthogonal Latin square pair:

```
1  2  3  4  5          1  2  3  4  5
2  3  4  5  1          3  4  5  1  2
3  4  5  1  2          5  1  2  3  4
4  5  1  2  3          2  3  4  5  1
5  1  2  3  4          4  5  1  2  3
```

If we superpose the two matrices, we get

```
1,1  2,2  3,3  4,4  5,5
2,3  3,4  4,5  5,1  1,2
3,5  4,1  5,2  1,3  2,4
4,2  5,3  1,4  2,5  3,1
5,4  1,5  2,1  3,2  4,3
```

There is no repeating pair, all the 25 pairs are distinct.

Write a function in **`ortho_Latin`** in Matlab that takes two n by n arrays A and B as input and returns **`true (false)`** if the pair (A, B) forms an orthogonal Latin square pair. Your program should first check that the inputs are square matrices, then check that they have the same dimensions. Next it should check that A and B are both Latin squares. Finally, check that they are orthogonal.

5) Let C be a cell array whose members can be cell arrays or numbers. Example: {{2, {4, 5, 6}, {{7, 8}, 1, 11}, 12}, {15, {21, 1}}}. Note that cell arrays can be arbitrarily deep in containing other cell arrays. You are to write a program **`get_numbers`** that takes such a cell array C as input and returns a vector containing all the numbers in the cell array (in the same order).

Example:

```
>> get_numbers({{2, {4, 5, 6}, {{7, 8}, 1, 11}, 12}, {15, {21, 1}}})

ans =

    2     4     5     6     7     8     1    11    12    15    21     1
```

Hint: The best way to solve this problem is to use recursion.