Assignment 4 - Computer Programming (0455-3391-01)

http://www.cs.tau.ac.il/~efif/courses/ComputerProgramming

Due: January 7, 2003

Problem 1

Write two programs that read input lines and print each line backwards. A line terminates with the '\n' character. The entire input terminates with the EOF.

1. For the first program write a function that reverses its input string in place

   ```
   void reverse(char line[], int len);
   ```

   Assume the length of a line is at most 80 characters.

2. For the second program, don't make any assumptions. Instead, use recursion.

Problem 2

Write a program that reads an input line, and determines whether it is a palindrome or not.
A palindrome is a string that is the same when read backward or forward.
Disregard spaces, non alphabetic and numeric characters, and ignore case distinctions. For example, the following are palindromes:

> Was it a rat I saw?
> Dennis and Edna sinned.
> step on no pets
> A Fool, A Tool, A Pool; LOOPALOOTALOOFA!
> Madam, I'm Adam.
> A man, a plan, a cat, a ham, a yak, a yam, a hat, a canal-Panama!

Problem 3

1. Write a program that merges two sorted sequences of numbers. In particular, write a function that accepts 5 parameters

   ```
   void merge(int res[], int seq1[], int len1, int seq2[], int len2)
   ```

   where *seq1* and *seq2* are the input sorted arrays of integers in ascending order, *len1* and *len2* are the lengths of *seq1* and *seq2* respectively, and *res* is the resulting merged array. Its length is the sum of *len1* and *len2*.

2. Write a program that sorts a sequence of integers using the merge function above. Use the following recursive function.

```
void sort(int numbers[], int len)
{
  int halfLen;

  if (len == 1) return;
  halfLen = len / 2;
  sort(numbers, halfLen);
  sort(&numbers[halfLen], len - halfLen);
  merge(numbers, numbers, halfLen, &numbers[halfLen], len - halfLen);
}
```

You may assume that no more than 100 integers are provided as input.

Problem 4

Given two equal-length strings of DNA, we define their *homology* as the percentage of matching nucleotides. For example, the *homology* of the following two strings:

```
GATAATTTATAATAT
GCTTTTTTATACTAA
```

is $\frac{10}{15} = 66\%$, since there are 10 matching nucleotides.
Write a program that reads a pair of strings and computes their *homology*.