Cs352 — Homework #3

February 10, 2004

Turnin ID: *cs352_assg3*

Turnin files: all .c files.

(PS: For example, you may turnin multiple files by UNIX command "turnin *cs352_assg3* yourfile1 yourfile2 yourfile3 yourfile4". You may also turnin one file at a time by "turnin *cs352_assg3* *file_you_want_to_turnin*". A later turnin file will overwrite the old file which has the same filename. So if you turnin the same file multiple times, we only receive the last version you turned in. To see a list of the files you turned in, you may use the command "turnin -ls *cs352_assg3*". For help information about turnin program, please use the command "turnin -h". If you still have questions about turnin, please either stop by TA's office hours or email TA's.

In this homework documentations will be 20% of the final grade. Please refer to *www.cs.arizona.edu/classes/cs352/spring04/index.html* for guidelines on proper coding guidelines.

Due Time: 2/17/04 (3:30PM)

1. Write a C program (named ElementUniqueness.c) that reads (using scanf()) a list of integers from stdin and prints them (using printf()). The program should exit once the integer '-1' is entered. After each element that is read (excluding the terminating '-1') the program should print the string "YES" if the last element read appeared already in the sequence. Otherwise it should print "NO". You may assume that the number of elements is no more than 100. Use pointer arithmetic as much as possible. For example

   lec:xxx > ElementUniquness
   1 7 3
   OUTPUT:
   NO NO NO
   -23 88 7 3 103 88 7 -1
   OUTPUT:
   NO NO YES NO YES YES

2. Write a function called RecursiveStrcpy whose prototype is `void RecursiveStrcpy(char *s1, char *s2)`. Demonstrate this function in a program called `RecursiveStrcpy.c`,

which repeatedly reads a string from the input (stdin), into the string s1, calls this function that recursively copies s1 into s2, and prints s2. The program exits once the string "end" is entered as input (use the function `strcmp` which is defined in `string.h`).

3. A function called `ReverseStringCpy`, whose prototype is `void ReverseStringCpy(char *s1, char *s2)` that copies into s2 the reverse string of s1. So for example, if s1="Hello", and `ReverseStringCpy(s1, s2)` is called, then s2="olleH". Demonstrate your function using the program `ReverseStringCpy.c`. The program should read zero or more strings given to it as parameters in the command line, and print the reverse name of each of them in a separate line, in the order they appear. For example, if the user typed
ReverseStringCpy Hello world what a nice day
OUTPUT: *olleH*
dlrow
thaw
a
ecin
yad

4. Write a program called `MatrixMultiplication.c` that reads two matrices (lets call them A and B), each of size $N \times N$, and prints the matrix $A \cdot B^t$ which is the product of the first input matrix with the transpose of the second input matrix. The input to both matrices should be given in a "raster-scan" fashion, i.e., scanning each row from bottom to top. Each element in the input matrices is a positive integer smaller than 52. The output should be printed as a matrix, with all numbers properly aligned. The constant N is defined at the beginning of your program, and should be defined as 3. Your should use pointer arithmetic as much as possible. Your program should be as efficient as possible. So use incrementing of pointers (p++) as much as possible. Use comparison between pointers (i.e. `for( p=p1; p<=p2; p++)` as much as possible, to check exiting conditions from loops. Though not checked in this example, think that N is a huge number. In the example below, the program output is printed in `Typewriter` font.

```
Enter the elements of the matrix A:
1 2 3   4 5 6   7 8 9
Enter the elements of the matrix B:
0 0 2   0 1 0   1 0 0
The output of the multiplication of A by the transpose of B is
7 8 18
4 5 12
1 2 6
```

To make sure we are all on the same page: If A and B are two $N \times N$ matrices, then the transpose of B, noted B^t, is the also an $N \times N$ matrix such that for every $0 \leq i, j < N$,

$$B^t[i][j] = B[j][i] \ .$$

If we define a new matrix C such that $C = A \cdot B^t$, then C is also an $N \times N$ matrix, and for every $0 \leq i, j < N$

$$C[i][j] = \sum_{k=0}^{N-1} A[i][k] \cdot B[j][k] \ .$$

5. Compile and run the program `queenSolver.c` in *www.cs.arizona.edu/classes/cs352/spring04/demos/*. Try to understand how it works. Write a 10-lines essay about the importance of proper documentations, and how it is related to what you did last summer.