ENEE 140 Lab 11

Lab instructions

This handout includes instructions for the recitation sessions on Wednesday and Friday. **Submit the homework** as indicated below. To prepare for the next lecture, complete the **reading assignment** and try to solve the **weekly challenge**.

Homework

Due: ... at 11:59 pm.

Create two programs by following the instructions below. Submit them using the following commands:

```
submit 2016 spring enee 140 AAAA 11 word_count.c
submit 2016 spring enee 140 AAAA 11 file_interleaving.c
```

Note: you must replace `AAAA` with your own section number (0101, 0102, etc.)

1 Word count

Write a program, called `word_count.c`, that receives one argument from the command line:

```
word_count file1.txt
```

This operation implements the functionality of the UNIX `wc` command. Your program should print out the following information about the input file:

- The number of lines
- The number of words
- The number of characters

Each line has no more than 100 characters. A line ends with the `'\n'` character except that the last line of the file ends with EOF. A word is defined as a string of letters (upper and lower case) and digits (0-9), starting with a letter or digit and ending before the first character that is not a letter or digit. A word has no more than 30 characters. For example, `J.K. Rowling's Harry Potter` will be considered as 6 words: `J`, `K`, `Rowling`, `s`, `Harry`, `Potter`.

2 File interleaving

Write a program, called `file_interleaving.c`, that receives three arguments from the command line:

```
file_interleaving file1.txt file2.txt file3.txt
```

Your program should open files `file1.txt` and `file2.txt` for reading and to create file `file3.txt` as follows:

- The first line of `file3.txt` is the first line of `file1.txt`
- The second line of `file3.txt` is the first line of `file2.txt`
- The third line of `file3.txt` is the second line of `file1.txt`
- The fourth line of `file3.txt` is the second line of `file2.txt`
- ...

When one input file reaches the EOF, the remaining lines in the other file should be copied to the output file and the program terminates. Your program should print appropriate error messages if fewer than 3 file names are provided on the command line or if the files cannot be opened.

Reading assignment

K&R Chapter 5.7. Re-read chapters 3.5, 3.7.

Weekly challenge

Write a program that places eight queens on a chess board so that not two queens threaten each other. A chess board has 8 x 8 squares, and a queen threatens the chess pieces on the same row, the same column, or the same diagonal.

There are $\binom{64}{8} \approx 4.4E9$ possible placements of 8 queens on a chess board, so trying them all would take a long time. Instead, try to place the queens incrementally, one by one. You can only have one queen in any row of the board, so after you have placed a queen you can move to the next row. When placing a new queen, check if the position doesn't threaten any of the queens already placed. If it does not, move to the next row; if it does, try the next column. If you have tried all the columns in the current row and you cannot place the queen in any of them, this means that the current queen placements cannot lead to a solution. In this case, return to the previous row and try to move the queen there to the next column. This strategy is called backtracking.

You can use the following template (also available in the GRACE class public directory, at `public/challenges/week11`):

```
/*
 * eight_queens.c  --
 *
 *   Place eight queens on a chess board so that not two queens threaten
 *   each other. A chess board has 8 x 8 squares, and a queen threatens
 *   the chess pieces on the same row, the same column, or the same diagonal.
 */


#include <stdio.h>

struct position {
    int row;
    int col;
};


int is_valid(struct position queens[], int row, int col);
void print_chess_board(struct position queens[]);


int
main()
{

    return 0;
}

void
print_chess_board(struct position queens[])
```

```
{
    int i,j;

    for (i=0; i<8; i++) {
        for (j=0; j<8; j++) {
            if (queens[i].col == j)
                printf("X ");
            else
                printf("o ");
        }

        printf("\n");
    }
}
```

The weekly challenge will not be graded. However, if you manage to solve it, you may submit it for extra credit. The deadline for submitting your solution to the weekly challenge is **Monday at 11:59 pm**. To be eligible for extra credit, you must implement correctly **all but two** of the weekly challenges. You can submit your program from a GRACE machine using the following command (replace `AAAA` with your section number):

```
submit 2016 spring enee 140 AAAA 1011 eight_queens.c
```