ENEE 140 Lab 10

Lab instructions

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

1 String reverse

Write a C program that reads in a string of no more than 15 characters (no white space) and then print it out backwards. For example, on input

```
Floatingpoint
```

your program should output

```
tniopgnitaolF
```

Try to implement this both using a for loop and a while loop.

Hint. The length of the input string is unknown, so you need to figure that out first. Remember that every string ends with the end of string character `'\0'`.

Homework

Due: Friday at 11:59 pm.

Create two programs by following the instructions below.

Create a .zip archive containing your programs, then log into Elms, click on Gradescope in the course menu, then go to the relevant assignment to submit your work.

1 Multiplication table

Use nested loops to print out the multiplication table as follows, use 2 spaces for the product and leave a tab between the equations:

```
1x1= 1 1x2= 2 ... 1x9= 9
2x1= 2 2x2= 4 ... 2x9=18
...
9x1= 9 9x2=18  ... 9x9=81
```

You can start from the `multiplication_table.c` program from the class public directory on GRACE and modify it so that it prints out the table in the following three formats:

- rotate the table

  ```
  1x1= 1 2x1= 2 ... 9x1= 9
  1x2= 2 2x2= 4 ... 9x2=18
  ...
  1x9= 9 2x9=18  ... 9x9=81
  ```

- only the lower triangle half

  ```
  1x1= 1
  2x1= 2 2x2= 4
  3x1= 3 3x2= 6 3x3= 9
  ...
  9x1= 9 9x2=18  9x3=27 ... 9x9=81
  ```

- only the upper triangle half

2 Generate random permutations

Write a complete program, called `permutations.c` to generate a random permutation of numbers from 1 to 10. That is, rearrange the 10 numbers randomly and print out the result. You need to use `rand()` and `srand()`. For example, all the following are valid:

```
 1  3  5  2  10  9  6  7  4  8
 2  8  9  1  10  7  4  5  3  6
10  6  3  8   1  5  7  9  2  4
```

Also keep track how many times you have called `rand()`; you should only call `srand()` once.

Hint. One idea to solve the optional question (but it may use a lot of rand() calls):

Step 1. generate a random number r between 1 and 10
Step 2. check whether r had been used or not
Step 3. if yes, go back to Step 1
Step 4. else

Step 4.1. print out r
Step 4.2. if all 10 number are generated stop
Step 4.3. else go back to Step 1

To implement this simple idea, (1) you will need an array to store the numbers generated for the check in step 2. (2) r needs to be put into the array at step 4.1 (3) You need to update the counter after step 1 to track the times we use rand(). You can improve step 2 by using another array `flag[10]` to track whether number `i+1` has been used or not. Set `flag[i] = 0` initially for all i and update `flag[r-1] = 1` when a new number `r` is generated.

(Challenging): Can you solve this by using only 9 `rand()` calls?

Reading assignment

K&R Chapters 7.1, 7.5, 7.6, 7.7, B1. Re-read chapters 7.2, 7.4.

Weekly challenge

Write a program that reads several file names from the command line, concatenates these files, and prints the result to the standard output.

You can use the following template (also available in the GLUE class public directory, at `public/challenges/week10`):

```c
/*
 * cat.c
 *
 *  Read several file names from the command line.
 *  Concatenate these files and print the result to the standard output.
 */

#include <stdio.h>


// Copy the content of one file to another; assume that the files are open.
// Return the number of characters copied
int
filecopy(FILE *in, FILE *out)
{
}

int
main(int argc, char *argv[])
{

        return 0;
}
```

```c
/*
 * cat.c
 *
 *  Read several file names from the command line.
 *  Concatenate these files and print the result to the standard output.
 */

#include <stdio.h>


// Copy the content of one file to another; assume that the files are open.
// Return the number of characters copied
int
filecopy(FILE *in, FILE *out)
{
}

int
main(int argc, char *argv[])
```

```
{

        return 0;
}
```

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**.

Submit it by going to the relevant assignment in Gradescope.