Week 3: In-Class Exercises

Resource: http://blue.smu.edu.sg/cs101/ice3-resource.zip

1. **[Difficulty: *]** Write a program that calculates the money you will get by depositing a certain amount of money into a savings account and leaving it in the bank for a period of time to draw interest after the period. The following formula is used:

$$A = P(1 + \frac{r}{n})^{nt}$$

A: The future value of the investment/loan, including interest
P: The principal investment amount (the initial deposit or loan amount)
r: The annual interest rate (decimal). Example, 0.01 for 1%
n: The number of times that interest is compounded per year (1 - annually, 12 - monthly)
t: The number of years the money is invested or borrowed for

```
Initial Deposit(P):1000
Annual Interest Rate (in %, r):1
Annual Compound Frequency (1 to 12, n):1
Num years (t):5
Amount after 5 years is 1051.01
```

Hint: Use the `pow()` function in the math library.

2. **[Difficulty: *]** Write a program that prompts the user for a positive number(n) and prints all the numbers from 1 to n separated by a comma (',').

```
Enter n:10
1,2,3,4,5,6,7,8,9,10
```

3. **[Difficulty: *]** Write a program that prompts the user for 5 numbers, and displays the smallest and largest integers input so far.

```
Enter n:4
Largest so far == 4. Smallest so far == 4.

Enter n:2
Largest so far == 4. Smallest so far == 2.

Enter n:7
Largest so far == 7. Smallest so far == 2.

Enter n:5
Largest so far == 7. Smallest so far == 2.

Enter n:9
Largest so far == 9. Smallest so far == 2.
```

4. **[Difficulty: *]** The factorial of a number is the product of all the integers from 1 to that number. Factorial is not defined for negative numbers, and the factorial of zero is one.

```
0! = 1
1! = 1
2! = 1 * 2 = 2
3! = 1 * 2 * 3 = 6
```

Write a program that prompts the user for a number and prints the factorial of the number.
Note: You can assume that the user will always enter a number.

```
cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:0
0! = 1

cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:2
2! = 2

cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:-1
Invalid input.
```

Hint: Try this program, and enter the values 0 and 1. When n is 0, is line 13 executed?

```
01  #include <stdio.h>
02
03  int main(void) {
04      int n;
05      printf("Enter n:");
06      scanf("%d", &n);
07
08      if (n == 0) {
09          printf("zero\n");
10          return 0;  // 0 for success, 1 for failure
11      }
12
13      printf("%d\n", n);
14  }
```

5. **[Difficulty: **]** The "Fizz-Buzz test" is an interview question designed to help filter out 99.5% of programming job candidates who can't seem to program. Write a program that prompts the user for a number, n. It then prints the numbers from 1 to n. But for multiples of three, print 'Fizz' instead of the number, and for the multiples of five, print 'Buzz'. For numbers which are multiples of both three and five, print 'FizzBuzz'.

```
Enter n:15
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz
```

6. Write code using a single `for`-loop (**no NESTED loops**) without `if` statements to produce each of the following outputs.

```
cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:3
--\|/---\|/---\|/-

9 4 1

123 234 345

AaBbCc

3 2 1 (sum: 6)

cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:5
--\|/---\|/---\|/---\|/---\|/-

25 16 9 4 1

123 234 345 456 567

AaBbCcDdEe

5 4 3 2 1 (sum: 15)
```

	size = 3	size = 5
A.[*]	--|/---|/---|/-	--|/---|/---|/---|/---|/-
B.[*]	9 4 1	25 16 9 4 1
C.[*]	123 234 345	123 234 345 456 567
D.[**]	AaBbCc	AaBbCcDdEe
E.[**]	3 2 1 (sum: 6)	5 4 3 2 1 (sum: 15)
F.[**]	1 21 321	1 21 321 4321 54321

OPTIONAL

7. Write code to produce each of the following outputs:

```
cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:3
***
***
***

***
**
*

123
12
1

  C
 B
A

* *
 *
* *

cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:5
*****
*****
*****
*****
*****

*****
****
***
**
*

12345
1234
123
12
1

    E
   D
  C
 B
A
```

```
*    *
 * *
  *
 * *
*    *
```

	SIZE = 3	SIZE = 4	SIZE = 5
A. [*]	*** *** ***	**** **** **** ****	***** ***** ***** ***** *****
B. [*]	*** ** *	**** *** ** *	***** **** *** ** *
C. [**]	123 12 1	1234 123 12 1	12345 1234 123 12 1
D. [**]	C B A	D C B A	E D C B A
E. [***]	* * * * *	* * ** ** * *	* * * * * * * * *

9. **[Difficulty: **]** In number theory, a perfect number is a positive integer that is equal to the sum of all its factors excluding itself.
   For example, 6 is a perfect number because the sum of its factors i.e. 1 + 2 + 3 = 6. Other examples of perfect numbers are 28, 496.

   Write a program in a file that takes in a positive integer. The program should then check if the number entered is a perfect number or not and print the result. You can assume that only positive integers are entered.

Sample Run 1:

```
Enter positive integer:28
28 is a perfect number
```

Sample Run 2:

```
Enter positive integer:96
96 is not a perfect number
```

10. **[Difficulty: **]** Write a program that prints ascending sequences of digits. The program is required to get inputs for the starting digit of the ascending sequence and count of sequences. Assume that the user enters positive integers for all inputs.

Some sample runs are shown below:

```
cwarror:/mnt/c/cs101/ice3$./a.out
Enter count of ascending sequence:5
Enter the starting digit:2
2 23 234 2345 23456

cwarror:/mnt/c/cs101/ice3$./a.out
Enter count of ascending sequence:3
Enter the starting digit:6
6  67 678
```

11. **[Difficulty: **]** A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8.... The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means, the nth term is the sum of (n-1)th and (n-2)th term. Write a program that prompts the user for a number and print the first n numbers in the Fibonacci sequence.

```
cwarrior:/mnt/c/cs101/ice3$./a.out
Enter n:8
0 1 1 2 3 5 8 13
```

12. **[Difficulty: **]** Write a program that prompts the user for the number of rows and columns, and displays a frame-like pattern.

```
cwarrior:/mnt/c/cs101/ice3$./a.out
rows>4
```

```
cols>4
****
*  *
*  *
****

cwarrior:/mnt/c/cs101/ice3$./a.out
rows>3
cols>1
*
*
*

cwarrior:/mnt/c/cs101/ice3$./a.out
rows>2
cols>5
*****
*****
```

13. **[Difficulty: **]** Write a program that prints the Christmas tree. The Christmas tree is drawn as an isosceles triangle over a stem of height 5 always.

    As an example, for a height of 11, the Christmas tree is drawn like this:

01						*					
02					*	*	*				
03				*	*	*	*	*			
04			*	*	*	*	*	*	*		
05		*	*	*	*	*	*	*	*	*	
06	*	*	*	*	*	*	*	*	*	*	*
07						*					
08						*					
09						*					
10						*					
11						*					

    Prompt the user for the height of the Christmas tree. If the height entered by the user is less than 10, give an appropriate message and exit the program.

```
cwarrior:/mnt/c/cs101/ice3$./a.out
Enter the height of the tree, greater than 9:10
    *
   ***
  *****
 *******
*********
    *
    *
    *
    *
    *

cwarrior:/mnt/c/cs101/ice3$./a.out
Enter the height of the tree, greater than 9:5
Enter a number greater than 9
```

14. **[Difficulty: **]** This question introduces you to the break and continue keywords. When dealing with loops, you can use the break and continue keywords to get out of the loop.
    a. When **break** is encountered inside any loop, control is passed to the first statement after the loop. A break is usually associated with an if. The keyword break, breaks the control only from the loop in which it is placed.
    b. When continue is encountered inside any loop, control is passed to the beginning of the loop. A continue is usually associated with if Statement.

(a) Consider the following code:

```
01  #include <stdio.h>
02
03  int main(void) {
04      for (int i = 0; i < 3; i++) {
05          printf("A\n");
06          for (int j = 0; j < 3; j++) {
07              printf("B\n");
08              if (i == 1 || j == 1) { //breaks the loop when i or j = 1
09                  break;
10              }
11              printf("i: %d, j: %d\n", i, j);
12          }
13          printf("C\n");
14      }
15  }
```

Trace through the code manually and determine what will be printed out. Then compile and execute it to check if the output matches your expected one.

(b) Change line 9 to

```
continue;
```

Trace through the code to predict the output again.