

## CSE1301 Practical Session 4

### Input/Output, Selection and Iteration (10 marks)

You should complete **at least** the preparation question before the prac, but it is recommended that you make substantial progress in the other questions too.

**Coding style and documentation:** You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

#### PART 1 (3 marks): Iteration

- a) **[1 marks]** Write an algorithm that receives as input a positive integer  $n$  ( $n \leq 9$ ), and prints out a triangular array such that the first row has one element whose value is 1, the second row has two elements whose value is 2, and so on, up to the last row which has  $n$  elements. For instance, if  $n=7$ , the following triangle should be printed:

```

1
22
333
4444
55555
666666
7777777

```

Implement your algorithm in a C program, and test your program on a variety of inputs. Make sure your program works for odd and even inputs, and checks for negative inputs.

- b) **[2 marks]** Now modify your algorithm so that it takes two inputs:
- a value  $n$  that represents the number of elements in the longest row of a triangle.
  - a character  $c$  to be printed in place of each element.

Your algorithm should produce an inverted isosceles triangle of characters  $c$ . For example, for  $n=7$  and  $c='*'$ , the resulting triangle should be as follows:

```

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

```

Implement your algorithm in a C program, and test your program on a variety of inputs. Make sure your program works for odd inputs, and checks for even and negative inputs.

#### PART 2 (3 marks) [PREPARATION QUESTION]: Selection

The Couch Potato™ video shop has three rental rates (GST included):

Type of Video	Rent per tape
Overnight	\$7.00
Three-day	\$5.00
Weekly	\$3.00

To encourage customers to rent more videos during weekdays (Monday to Thursday), the shop offers the following promotions:

- **Promo 1 (Monday, Tuesday):** rent 2 overnight videos, and you get 1 three-day video free
- **Promo 2 (Wednesday, Thursday):** rent 2 three-day videos, and you get 1 weekly video free
- There are no promotions during weekends (Friday to Sunday)

For example, if a customer is renting 3 overnight videos and 2 three-day videos on Monday, the price for the customer is as follows:

- 3 overnight \_ \$7.00 each = \$21.00, and 1 three-day free (you save \$5.00)
- 1 three-day \_ \$5.00 = \$5.00
- total cost: \$26.00 (\$5.00 savings)

Your task is to write a C program for the shop's cash register. The program will ask for the day of the week (to be entered as a single character), and for the number of overnight, three-day and weekly videos the customer is renting. It will then print the total cost for the customer and the amount saved.

### Steps

1. **[1 mark]** Write a simple C program to input the day of the week, and the number of overnight, three-day and weekly videos the customer is renting. Compile this program, and print out the input values to ensure that they are read correctly.  
**Note:** since the day of the week is indicated by a single character, you will need to define a set of characters, e.g., 'm' for Monday, 't' for Tuesday, and 'h' for Thursday.
2. **[1 mark]** Update your program so that it also computes the total cost of renting the tapes, assuming the shop does not offer any promotions. Compile this program, and print out the total cost to ensure that it is computed correctly.
3. **[1 mark]** Update your program so that it takes Promo 1 and Promo 2 into account. Test your program to ensure that the total cost and total savings are computed correctly. **Hint:** use integer division to calculate the savings obtained from a promotion. Note that a promotion could apply more than once (e.g. if you rent 4 overnight videos).

### Notes:

- **Marking:** You are required to keep a copy of your program's version at each step. You should test and document your program as you go along.
- To make your program easy to update later in case of changes, you should store rental rates and promotion conditions in constants.

### PART 3 (4 marks): Iteration, Input/Output

The value of  $e^u$  is defined by the following sum:

$$e^u = \sum_{n=0}^{\infty} \frac{u^n}{n!}$$

- a) **[2 marks]** Write a program which calculates the value of  $e^u$ , and compare its output to the result of calculating  $e^u$  directly (using the C library function `exp`). Your program should
  - Receive two inputs: *limit* and *u*.  
*limit* is a number which replaces  $\infty$  in the above formula.  
*u* is the exponent to which we want to raise  $e$ .
  - For each term in the sum, calculate the value of  $u^n$ , then calculate the value of  $n!$ , then the result of dividing one by the other. Add the terms in the sum together to get the final result.
  - Print two results: one obtained by using the method above and one obtained from the direct calculation. Use a separate `printf` statement for each result.
- b) **[1 mark]** Run your program with different values for *limit* and *u*.

- test your program with values of limit of 5, 10, 20 and 30. Observe how the change in *limit* affects the difference between the results obtained by your program and the directly obtained results.
- Test your program for values of  $u$  of 1, 2 and 3.
- Record your results in the following table:

Value of $u$	Value of <i>limit</i>			
	5	10	20	30
1				
2				
3				

- c) **[0.5 mark]** Run your program with different precisions for the variable that stores the result of the above formula: `float`, `double` and `long double`. How do these precisions affect the result? Make sure you use the appropriate conversion specifier when printing the value of this variable for each of these precisions.
- d) **[0.5 mark]** Examine the effect of using different types of conversion specifiers in your `printf` statements. In particular, for the `long double` precision, consider the effect of using `%f`, `%Lf`, `%.8f` and `%.8Lf` for both `printf` statements. How do these conversion specifiers affect the results?

**Note:** Use the library functions `exp` and `pow` to perform the calculations required in your program. These functions are found in the C library `math.h`.

**Additional Component (2 bonus marks):**

- a) **[1 mark]** Write an algorithm that modifies Part 2(b), so that the triangle of characters  $c$  is printed right-side-up. Implement your algorithm as a C program.
- b) **[1 mark]** Can you think of a way to avoid the problems encountered in Part 3? (**Hint:** there is way to calculate the value of each term in the series without using either factorials or exponentiation! Can you see how to do it?)