COMP2401 - Assignment #1

(Due: **Friday, Sep 20, 2019 @ 6pm**) – updated 4:30pm on Sep 10

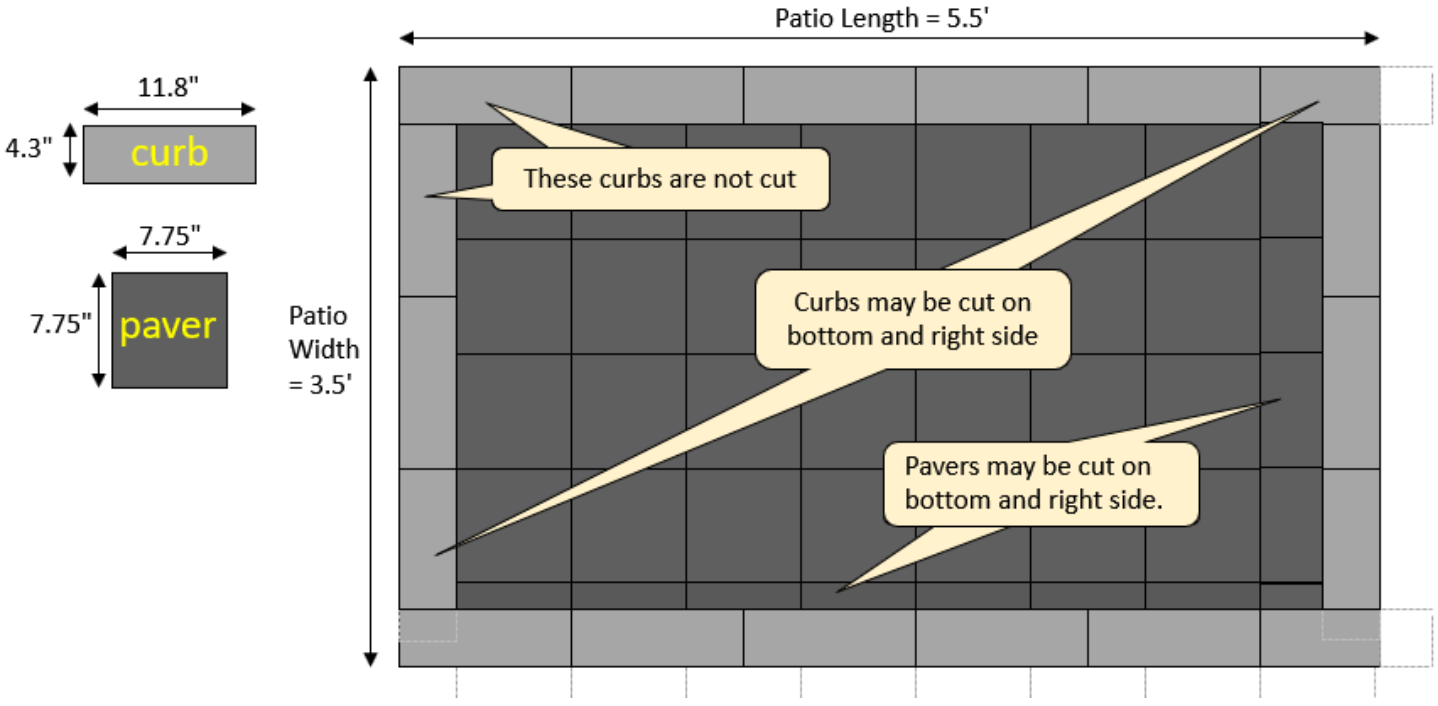
In this assignment, you will create some simple programs in C to get used to the language (i.e., control structs, variables, arrays), the Linux environment, and the compiling/running process.

ALL of your code for both questions must be written neatly with proper indentation and have a reasonable amount of comments. If you do not, you will be losing multiple marks.

---

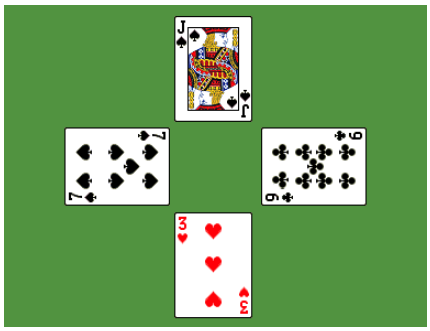
(1) Patio Stones

Assume that you want to buy some paver stones and curbs to build a patio in your backyard. Each paver stone is **7.75** inches x **7.75** inches square. You also want to put some concrete curb pieces around the outside of border of your patio. Each curb is **4.3** inches wide x **11.8** inches long. Write a program called **patioBuilder.c** that prompts the user for the desired patio width (in feet) and the length (also in feet). When getting the input, if the width or length is smaller than 24 inches, then quit the program with a message indicating that the patio dimensions are not big enough. Otherwise, the program should determine (and display) the number of concrete curbs and the number of paver stones that will need to be purchased in order to complete the patio, assuming that each time a curb or paver stone is cut, we cannot use the leftover piece. The curbs must be placed so that the first on the topleft and the first on the left side are not cut, as shown below. Assume a cost of **$2.48** for each curb and **$3.90** for each paver stone. Your code should display the number of curbs and pavers that need to be purchased as well as the cost for the pieces, HST amount and final cost. Make sure to define appropriate constants in your code, as this represents proper coding style.



(2) Card Hands

Write a program called **cards.c** that simulates some card game logic by comparing the cards from 4 people and determining which person has the best card. The program **MUST** work as follows:



1. Each card must be represented by exactly two **chars** representing a ***rank*** and a ***suit***. The possible ***rank*** options are: '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A'. ('T' stands for 10) The possible ***suit*** options are: 'H', 'D', 'S', 'C'. So **6H** represents the "6 of hearts", **JC** represents the "Jack of Clubs" etc…

2. You MUST write a function called **isValidRank(char c)** which determines if the given character is one of the ranks mentioned above. It should return a **char** with a value of **1** if the character is a valid rank and **0** otherwise. Lowercase letters are not valid.

3. You MUST write a function called **isValidSuit(char c)** which determines if the given character is one of the suits mentioned above. It should return a **char** with a value of **1** if the character is a valid suit and **0** otherwise. Lowercase letters are not valid.

4. You MUST have a function called **getTrump()** that returns a **char**. It should prompt the user for a ***trump*** suit, which must be 'H', 'D', 'S' or 'C'. It should be robust, in that any invalid input is not accepted. It should only return from the function when a valid suit has been entered, and it must make use of the **isValidSuit()** function. For any invalid entry, an appropriate error message should be given. Blank entries are invalid and so are lowercase letters.

5. The **main** function should first get the trump suit, by calling the above function. It should then enter an **infinite loop** to do the following: (1) ask for 4 cards from the user, (2) display the 4 cards entered, (3) determine and display which player wins the round (i.e., which one has the "best" card). These steps will be explained below.

6. When entering the cards … your code should be robust and handle any character input, just like you did in the **getTrump()** function. For each of the 4 cards entered, your code should allow the user to enter two characters and then press enter. You should check each character individually to make sure that the rank and suit are valid (use the functions you wrote earlier). If either are not valid (or a character is missing), then an appropriate error message should be displayed and the code should keep prompting until a valid card is entered before moving on to get the next player's card. Here is an example of what you should do →

```
Player 1: Enter card rank and suit (e.g., 2S, TC, KD)
RC
Invalid card, please re-enter
Player 1: Enter card rank and suit (e.g., 2S, TC, KD)
4F
Invalid card, please re-enter
Player 1: Enter card rank and suit (e.g., 2S, TC, KD)
6
Invalid card, please re-enter
Player 1: Enter card rank and suit (e.g., 2S, TC, KD)
H
Invalid card, please re-enter
Player 1: Enter card rank and suit (e.g., 2S, TC, KD)
JC
Player 2: Enter card rank and suit (e.g., 2S, TC, KD)
6S
Player 3: Enter card rank and suit (e.g., 2S, TC, KD)

Invalid card, please re-enter
Player 3: Enter card rank and suit (e.g., 2S, TC, KD)
9d
Invalid card, please re-enter
Player 3: Enter card rank and suit (e.g., 2S, TC, KD)
9D
Player 4: Enter card rank and suit (e.g., 2S, TC, KD)
5H
```

7. Once 4 valid card entries have been entered, the 4 cards should be displayed like this:

```
JC, 6S, 9D, 5H
```

8. You must then determine which card wins the round. That is, which player has the best card. To do this, you must follow these rules:

   - A card which is of the **trump** **suit always beats** a card that is a non-trump suit.

   - If two cards have the **same suit**, the one with the **higher rank is better**. 'A' is the highest rank and '2' is the lowest.

   - The card played by player 1 is called the "***suit led***". If no other player has a higher ranking card of the same suit as the ***suit led***, and no other player has the **trump** suit, then player 1 has the best card and wins.

9. In your **main** function, if the rank of the first (or any) player is a '**.**' character, then the program should quit. The TA's will need this functionality in order to test your program.

IMPORTANT SUBMISSION INSTRUCTIONS:

Submit all of your **c source code** files as a single **tar** file containing:

1. A **Readme** text file containing
   - your name and studentNumber
   - a list of source files submitted
   - any specific instructions for compiling and/or running your code
2. All of your **.c source** files and all other files needed for testing/running your programs.
3. Any output files required, if there are any.

The code **MUST** compile and run on the course VM, which is **COMP2401-F19**.

- If your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

- You WILL lose marks on this assignment if any of your files are missing. So, make sure that you hand in the correct files and version of your assignment. **You will also lose marks if your code is not written neatly with proper indentation and containing a reasonable number of comments.** See course notes for examples of what is proper indentation, writing style and reasonable commenting).
