Large Program 1
The Toothpick Game

COP3223C Introduction to Programming with C Fall 2022
Dr. Andrew Steinberg

Background Story of this Assignment

You are about to implement your first large program in this course. Large programs takes multiple concepts we have learned that will be applied in one problem that you will code in C. In this assignment, the primary objective is to apply your skills in user-defined functions, control flow, and conditions.

In this assignment, you are going to implement the toothpick game! If you haven't played the game, here are the general rules. A table contains 31 toothpicks. There are two players (in this assignment you and the computer) who will each take turns picking up either 1, 2, or 3 (cannot grab 4 or more toothpicks or even put back toothpicks) toothpicks off from the table. The objective of winning the game is to not be the last person to pick up the final toothpick. The player that picks up the last toothpick loses. Utilizing the concepts of user-defined functions, control flow, and conditions, you are going to implement this classic game. In this programming assignment, the user will always go first.

Due Date

The assignment is due on October 2nd at 11:59pm EST via Webcourses. **Do not email the professor or TAs your submissions as they will not be accepted!** This assignment is accepted late up to 24 hours with a penalty. Please see the syllabus for more information on this. Make sure to submit on time to get potential full credit. Make sure to also take into consideration the uploading time. In the past, students who are working last minute on the assignment sometimes run into uploading issues where their Internet may run slow, resulting in late submissions. The timestamp Webcourses uses for your submission will be applied and will be the final say. Please do not email the instructor or TAs saying your Internet was running slow. If the time is off by a second of the due date, then the assignment is considered late. Plan accordingly and start early!

Important! Read Carefully!

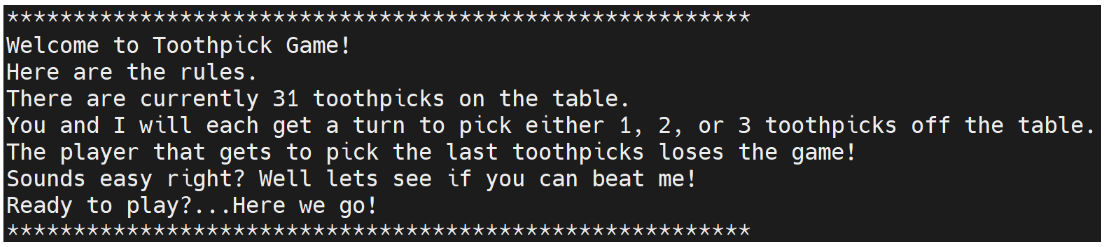
In this assignment, you are going to implement **6 user defined functions** to simulate the classic game.

```
void greeting(); //display welcome message to user
int playRound(int round); //play one round
int humanPick(); //retrieve the user's guess
int computerPick(); //computer makes its pick
int leftOnTable(int toothpicks, int taken); //calculate number of ↩
    toothpicks left
void winnerAnnouncment(int user); //overall winner of round announcement
```

You cannot add any additional functions or remove them. Modifying them in any way (name, parameters, etc...) will result in point deductions! Utilize them the way they are provided!
The function prototypes and partial definitions are provided for you in the file LargeProgram1_Skeleton.c for this assignment. Pay close attention to the comments of where you will fill in the missing code. Download the file and fill in the missing blanks to make the program run perfectly. Do NOT modify any of the code that was already provided for you! You can remove the comments that provide hints of missing code. Any modifications from the provided code will result in points being deducted! You are allowed to use the same message that is provided in the sample output for your program. You can also make modifications to the text of messages as long as the program follows all directions as stated in this assignment.
The following section will discuss the prototypes more in detail. Make sure that you name your C file *largeprogram1_lastname_firstname.c*, where lastname and firstname is your last and first name (as registered in webcourses). For example Dr. Steinberg's file would be named *largeprogram1_Steinberg_Andrew.c*. Make sure to include the underscore character _. If your file is not named properly, points will be deducted.

The Function Prototypes

This section will discuss the function prototypes.



```
********************************************************
Welcome to Toothpick Game!
Here are the rules.
There are currently 31 toothpicks on the table.
You and I will each get a turn to pick either 1, 2, or 3 toothpicks off the table.
The player that gets to pick the last toothpicks loses the game!
Sounds easy right? Well lets see if you can beat me!
Ready to play?...Here we go!
********************************************************
```

Figure 1: Sample output from the greeting function.

```
void greeting();
```

The greeting function will welcome the user to the game and explain the rules on the terminal window. See the figure 1 of what the function produces when invoked. This should be the first thing that is displayed when the program begins its execution.

```
int playRound(int round);
```

The `playRound` function simulates an entire round of the game. The function has one parameter of type int which represents the current round being played. Inside the function, a loop has been provided for you (do not modify the loop). You are going to fill in the code that simulates the entire round of the game. That means you will need think about how execute each turn between the user and computer (think about how you will call those other user defined functions). The function should display the number of toothpicks left on the table and allow the user to make a selection (hint think about humanPick function). Also, you will need to take into consideration if the user decides to cheat by taking the incorrect number of toothpicks. If so, the function should display message that tells the user that they are breaking the rules. In this case, make sure to let the user go again (hint think about using a condition). This also includes the scenario where the user might grab extra toothpicks when there is not enough on the table (example, user tries to grab 3, but there are only 2 on the table should not be allowed). If the user makes a valid selection, then the program should let the computer make a pick (call the computerPick function). After each player makes a valid pick, display the number toothpicks taken off the table. Once the table has 0 toothpicks, the function should terminate and return an integer representing who went last in picking up the final toothpick(s). **Note: This function has already been called for you in the skeleton file of the main function. You will just need to implement the definition of the function of what it is suppose to be accomplished. You were also provided some components of the code which is the while loop since loops haven't been discussed yet. Please note that anything inside the control structure of loop will be repeated. Make sure to think about the lines of code you want to see repeated in execution.**

```
int humanPick();
```

The `humanPick` function will ask the user how many toothpicks they want to take. The user will enter an integer value from the keyboard. They can only take 1, 2, or 3 toothpicks. After the user makes a selection, he function returns the value. **Important!** It is possible that the user can make an invalid choice (typing in the wrong number). However, you do not need to worry about a non numerical character being typed in by accident. If the user makes an invalid numerical choice, the program will inform the user by displaying a message to the terminal (that part is done inside of the playRound function). That means the invalid numerical number is also returned.

```
int computerPick(int choice, int leftover);
```

The `computerPick` function allows the computer to make a selection. The function has two paramters. The first paramter is a value that represents the number of toothpicks the user took in the last turn (1, 2, or 3). The second parameter represents the value of toothpicks left on the table. Now the computer has a secret strategy of selecting a number and it based on what the user does (how sneaky). Here is how the computer makes it moves. It chooses one of three options.

1. If there are more than 4 toothpicks left on the table, then the computer should take $4 - x$ toothpicks, where x is the number of toothpicks the user took in the previous turn.
2. If there are 2 to 4 (both inclusive) toothpicks left, then the computer should withdraw enough toothpicks to leave 1 left on the table for the user to select.
3. If there is 1 toothpick left on the table, then the computer takes the last toothpick and of course loses the round.

```
int leftOnTable(int toothpicks, int taken);
```

The `leftOnTable` function will simply calculate the number of toothpicks left on the table of a player removes them from the table. It has two parameters. The first parameter represents the number of toothpicks on the table and the second parameter represents the number of toothpicks taken based on the respective player’s turn. The resulting value is returned.

```
void winnerAnnouncment(int user);
```

The `winnerAnnouncment` function determines the overall winner of the round. The function has one parameters. The value represent the user who won. You will use this value to determine the winner. Based on the winner, you will display some of sort of message to the terminal. If you the user wins, then the message `You won!  I'll let you have this one.` (this is the computer talking to you) is displayed. If the computer wins, then the message `I won!  Haha better luck next time!` is displayed to the terminal. This function is invoked in the main function after the playRound function terminates.

The Skeleton File

For this large program, you are given a skeleton file with some code. The function prototypes and partial definitions are provided for you in the file `LargeProgram1_Skeleton.c` for this assignment. Pay close attention to the comments of where you will fill in the missing code. Download the file and fill in the missing blanks to make the program run perfectly. **Do NOT remove the loops that were provided for you.** Any loops removed from the provided code will result in points being deducted! You are allowed to use the same message that is provided in the sample output for your program. You can also make modifications to the text of messages as long as the program follows all directions as stated in this assignment.
You can find the skeleton file in this pdf along with the actual C file in the assignment page of webcourses. You will see the partial comment header, preprocessor directives, function prototypes, the main function, and one of the function definitions partially implemented. The main function provided has some missing items that you will need to fill in (comments are provided in skeleton). Some of you are probably curious about the for statement. This is a counting loop that will allow us to execute code a number of times without rewriting mulitiple times. This topic will be covered after Exam 1. There is nothing else that needs to be added for the main function. As for the partial definition of `playRound`, there will be code that you need to fill in. Now, some of you probably have not seen the while statement before. This is another loop we will cover after Exam 1. The while loop you see in the code is conditional. This means anything inside the control structure of the while loop will keep executing until the condition between the parenthesis has evaluated to false (note you should be able to recognize the expression from learning conditions). The code in the while loop's control structure will keep executing until there no more toothpicks left on the table. You will have to write code inside the control structure of the while loop to complete this function's definition. You will also have to write code outside the while loop's control structure as well. Look at the sample output provided and please ask the TAs or course instructor for clarification.

```c
//Name:
//Dr. Steinberg
//COP3223C
//Large Program 1 Skeleton

#define ROUNDS 3
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>

//void greeting(); //display welcome message to user
int playRound(int round); //play one round
//int humanPick(); //retrieve the user's guess
//int computerPick(int choice, int leftover); //computer makes its pick
//int leftOnTable(int toothpicks, int taken); //calculate number of toothpicks left
//void winnerAnnouncment(int user); //overall winner of round announcement

int main()
{
    //insert some code here that will greet the user

    for(int x = 0; x < ROUNDS; ++x)
    {
        int result = playRound(x + 1); //call playRound and assign result the value function↩
             returns

        //insert some code here that will determine the winner
    }

    printf("********************************************************\n");
    printf("Thank you for playing!\n");
    return 0;
}

int playRound(int round)
{
    printf("Welcome to a new round %d!\n", round);
    printf("You may go first!\n");

    int toothpicks = 31; //number of toothpicks to start with

    //you can insert code here

    //loop that keeps track of toothpicks until respective no more toothpicks left. we will ↩
        learn about conditional loops soon :)

    while(toothpicks != 0)
    {
        //insert code here that simulate the round properly based on assignment directions

        return 0; //terminates loop HOWEVER YOU WILL NEED TO CHANGE THIS WHEN BUILDING YOUR ↩
            PROGRAM. THIS WAS PUT IN THE SKELETON SO THAT THE INITIAL RUN ISN'T STUCK IN AN ↩
            INFINITE LOOP
    }

    return 0; //returns 0 HOWEVER YOU WILL NEED TO CHANGE THIS PART OF THE CODE TO MAKE THE ↩
        PROGRAM WORK PROPERLY! YOU DON'T WANT THE SAME VALUE RETURNED ALWAYS
}
```

Testing on Eustis

It is your responsibility to test your code on Eustis. If you submit your assignment without testing on Eustis, you risk points being deducted for things that may behave differently on your operating system. Remember, you cannot dispute grades if your code didn't work properly on Eustis all because it worked on your machine. The Eustis environment gives the final say. Plan accordingly to test on Eustis!!

Why was I not provided a Python Script File?

You are probably wondering why Dr. Steinberg didn't provide a python script to check your code like in the small programs. The reason is that Dr. Steinberg wants his students to enjoy the assignment without being told of what must be displayed to the terminal and matching exactly. For large programs, Dr. Steinberg allows students to change the text output to the terminal, however your program must perform the specific functionalities that is requested at least to receive potential full credit.

The Rubric

Please see the assignment page for the established rubric on webcourses.

Comment Header

Make sure you place a comment header at the top of your C file. You will use single line comments to write your name, professor, course/section number, and assignment. For example, Dr. Steinberg's header would be:

```
//Andrew Steinberg
//Dr. Steinberg
//COP3223C Section 1
//Large Program 1
```

Missing a comment header will result in point deductions!

The Solution Text File

I have provided a sample output of a working program of what is expected in webcourses. As I stated before, you are welcome to change the text output to the terminal, or you can keep it exact like in the sample.

Recommendations for Completing the Assignment

Here is the order of steps I would strongly consider when attempting the large program assignment. After each of these steps, see if your program builds/compiles successfully and performs the correct task. See if the output is what you were expecting.

1. Work on the greeting function. See if you can get the program to welcome the user to the game.
2. Work on the leftOnTable function. It is actually simple.
3. Work on the humanPick function. See if you can successfully collect the user's input and return. I would recommend testing this by writing a printf statement after the function is invoked.
4. Work on the computerPick function. Think about all possible scenarios that can happen based on the user's selection.
5. Work on the playRound function. This function is the heart and soul of the program. Think about the order of the steps taken in implementing the game. How does humanPick and computerWork help out? How would the leftOnTable function call work in this user-defined function? How would you terminate the function properly when the round is over and what value would you send back to the main function.
6. Work on the winnerAnnouncment function. Using the value passed, how are you able to determine who won?

Basic Rules for Completing The Assignment

Please make sure to follow these rules when completing the assignment.

- Please use all items that were presented to you in class and labs. Everything that was presented to you can help you compete the assignment.
- DO NOT COPY ANYONE'S CODE! This is cheating as stated in syllabus. This assignment is individual work.
- DO NOT SHOW YOUR CODE to classmates. The only group of people that can see your code are the TAs, ULAs, and Course Instructor. Follow the rules from the syllabus if you talk to other students. Otherwise you may be in cheating violations!
- Make sure to use good practice of writing code. Anything that is considered bad practice will result point deductions. Example GOTO statements should not be seen anyone's code. There is a reason I don't go cover them. They are extremely bad in practice to use and just awful.

Some Last bit of Advice

Here are some tips and tricks that will help you with this assignment and make the experience enjoyable.

- Do not try to write out all the code and build it at the end to find syntax errors. For each new line of code written (my rule of thumb is 2-3 lines), build it to see if it compiles successfully. **It will go a long way!**
- After any successful build, run the code to see what happens and what current state you are at with the program writing so you know what to do next! If the program performs what you expected, you can then move onto the next step of the code writing. If you try to write everything at once and build it successfully to find out it doesn't work properly, you will get frustrated trying find out the logical error in your code! **Remember, logical errors are the hardest to fix and identify in a program!**
- Start the assignment early! Do not wait last minute (the day of) to begin the assignment.
- Ask questions! It's ok to ask questions. If there are any clarifications needed, please ask TAs and the Instructor! We are here to help!!! Do not wait last minute to seek help as generally you could be waiting in a long line of fellow classmates who may need help. As stated previously start early and ask your questions right away!