CSE1222 Programming & Algorithms for Engineers **Autumn 2016**

Programming Assignment 5: Putting it all together

1 Setting up the Programming Environment

1. Create a new directory (folder) called `hw5` and move to that directory.

2. Copy the file `pickstones_solution.exe` from the directory `/class/cse1222/9643/hw5` into the current directory. Copy the file `pickstones_template.cpp` and rename it `pickstones.cpp` using the following command:

```
cp /class/cse1222/9643/hw5/pickstones_template.cpp pickstones.cpp
```

2 Work by Yourself

All lab and programming assignments are to be done by yourself. You may discuss labs or assignments with other students in the class but DO NOT LOOK AT ANYONE'S CODE OTHER THAN YOUR OWN. Needless to say, you should not share or copy anyone else's code.

3 Program Requirements

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the course notes. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments at the top of the program. The declaration of every variable should have a comment. You will use functions in your program.

Write a program for playing a variation of the Chinese game "Tsyan-shizi" called the game of NIM. Our version of the game starts with up to 20 rods and up to 10 stones in each rod. Two players take turns removing stones from the rods. On a player's turn, she chooses a rod and removes one or more stones from that rod. She can remove all stones from the rod. The player who takes the last stone from the rods (so that all rods are empty) wins.

Run the solution `pickstones_solution.exe` to play the game. Your program must produce exactly the same output as this solution.

DO NOT delete the code already given to you in the code template. You must use the code given to you in the template in your solution. The algorithm for the game is provided below. Most steps in the algorithm should be implemented as function calls. You need to implement the algorithm and the functions as described below. You may implement additional functions to help you in writing your solution. Though, in order to receive full

credit you must follow the instructions in this assignment and define these functions exactly as indicated.

You will implement the following algorithm in the **main** function. Each line in the algorithm will correspond to a function call. `Important!` Write your code `incrementally`. This means you should write each function one at a time, check that it compiles and runs properly, test it, and then implement the next function.

i. Prompt and read the number of rods

ii. Prompt and read the number of stones in each rod;

iii. Draw the rods with percentages;

iv. Display statistics;

v. **while** (*some rod is NOT empty*) **do**

vi. Prompt and read the next player's move (Prompt and read the rod to modify and the number of stones to remove from the rod);

vii. Remove the specified number of stones from the specified rod;

viii. **if** (*all rods are empty*) **then**

ix. Print a message congratulating the winning player.

x. **else**

xi. Redraw the rods with percentages;

xii. Display statistics;

xiii. Change to the other player;

xiv. **end**

xv. **end**

1. All functions should be written AFTER the `main` procedure.
2. All function prototypes should be written for each function and placed BEFORE the `main` procedure.
3. Each function should have a comment explaining what it does.
4. Each function parameter should have a comment explaining the parameter.

5. Your program MUST NOT use any global variables nor any global constants.

6. In addition to implementing the above algorithm in the **main** function, you must define the following constants and variables in the **main** function:

   - Define two constants to hold the maximum number rods (20) and the maximum number of stones per rod (10).

   - The list of rods will be stored in an array of integers, where each integer represents the number of stones in that rod. Define a pointer variable for this array (see the lecture notes on "Arrays and Pointers"). Also define an integer to hold the size of this array.

   - Define a variable that represents the current player (1 or 2).

7. Important: PLEASE READ! The following description provides specifications for the functions you will write that includes input parameters and return values. You are responsible for implementing these functions as described in these specifications in order to receive full credit. These functions may call additional helper functions that you define. You must determine good function names as well as good variable names for your function parameters and local variables.

   You must correctly identify whether a function parameter should be `passed by value` or `passed by reference`. In addition you must correctly determine whether a function parameter should be declared with the `const` specifier. Remember, `const` is used to indicate that the function parameter is not modified in the function (see the lecture notes). It may be applied to a parameter that is either passed by value or passed by reference. It allows code developers to better undertand your code and ensure that they (and you) adhere to your specifications if they want to modify the code.

8. Write a function to implement step (i) of the algorithm. Define the function to have one input parameter that holds the maximum number of rods (see the constant defined above) and return an integer. Prompt the user for the number of rods used in the game. If the number entered is not between 1 and the maximum number of rods then display a warning message and prompt again. This is called `input validation` on the user's input.

9. In the `main` function, allocate an array in the declared array pointer representing the list of rods (see description above) to have the same size as the number of rods returned from the function in step (i).

10. Write a `procedure` (this is a function that does not return a value and so is declared with a return type of `void`) that implements step (ii) of the algorithm. Define the procedure to have three input parameters: the array of rods, the size of the array, and the maximum number of stones per rod (see the constant defined above). This

procedure will traverse the array of rods and call the following helper procedure for each rod in the list.

- Write a helper procedure that will prompt the user for the number of stones to place on this rod. If the number of stones entered for the rod is not between 1 and the maximum number of stones allowed then display a warning message and prompt again. Define this helper procedure to have three input parameters: the array entry for this rod (use pass by reference), the array index for this rod, and the maximum number of stones per rod.

`Important`: Again, a `procedure` is a function that has no return value. You must define these functions as procedures by defining their return type as `void`.

11. Write a procedure to draw the rods to implement steps (iii) and (xi) of the algorithm. Draw each rod on its own row. In the row for rod i, there is a label "Rod i: " followed by n *'s where n is the number of stones in rod i. The procedure should output an empty line before and after the first and last rows, respectively. For instance if rod 0 has 3 stones, rod 1 has zero stones, and rod 2 has 8 stones, the procedure (and helper procedures) should output:

```
Rod  1: ***             (27.273%)
Rod  2:                 (0.000%)
Rod  3: ********        (72.727%)
```

Define the procedure to have two input parameters: the array of rods and the size of the array. First, calculate the total number of stones in the array of rods. Next, traverse the array of rods and call a helper procedure to display each complete row.

- Write the helper procedure to have three input parameters: the array index for this rod, the number of stones for this rod, and the total number of stones in the array of rods. The percentage displayed at the end of each row is the fraction of stones on this rod given the total number of stones in the array of rods (you just computed).

Format each row using `setw` in iomanip and avoid using hard coded spaces. When there are ten or more rods to display, your output must align the labels "Rod ?:" and "Rod ??:" appropriately, where "?" is a digit. Use `setw` to format the text between "Rod" and the ":". Using `setw`, format the *'s in a column of size ten that is left justified (look up the "left" specifier under iomanip). There needs to be five spaces between the column of *'s and the percentage. **Do NOT hard code these five spaces**. Instead, use `setw` to format the last column for the percentage. Look up how to use the "right" specifier in iomanip to help you solve this problem. Also, format the percentage to show three digits after the decimal place.

Decide if additional helper functions/procedures would be useful here.

12. Write a procedure to display the statistics for the list of rods that will be called in steps (iv) and (xii) of the algorithm. The statistics are 1) The rods with the smallest number of stones, 2) The rods with the largest number of stones, and 3) The average number of stones per rod taking into account only rods with stones. Define the procedure to have two input parameters: the array of rods and the size of the array. This procedure will call three helper procedures.

    - Write a helper procedure to display the rods with the smallest number of stones. This procedure will have two input parameters: the array of rods and the size of the array. Note that there may be more than one rod with the smallest number of stones.

    - Write a helper procedure to display the rods with the largest number of stones. This procedure will have two input parameters: the array of rods and the size of the array. Note that there may be more than one rod with the largest number of stones.

    - Write a helper procedure to display the average number of stones per rod, but only taking into account rods with at least one stone. The average must be formatted to display two digits after the decimal place.

13. Write a function which returns `true` (boolean) if all the rods have zero stones and returns `false` otherwise. Call this function to implement steps (v) and (viii) of the algorithm. Note that the same function will be used for both steps.

14. Write a procedure to implement step (vi) of the algorithm where the current player makes her next move. Define the procedure to have five input parameters: the array of rods, the size of the array, player id, which rod chosen by the player on this turn, and how many stones the player would like to remove from this rod. The player id is either the integer 1 or 2 to indicate which player is taking a turn. This procedure will perform two steps with the help of helper functions/procedures as specified below:

    1) Using the following helper functions and procedures, prompt and read for the rod from which to remove stones. If the player inputs an invalid rod id or chooses a rod with no stones then display a warning message and prompt again. These three helper functions/procedures will be called from this procedure. In addition, the last two helper procedures will call the first helper function.

    - First, write a helper function to have one input parameter, the player id. Prompt the player to choose a rod (rod id) and then return this value. Thus, this helper function has a return type that is NOT void. Do not perform any input validation in this function.

    - Next, write a second helper procedure to have three input parameters: the player id, the rod selected by this player, and the total number of rods (i.e., the size of the array of rods). This procedure will validate the rod selected by this player, i.e. the rod selected must be between 0 and n - 1 where n is the total number of rods. If the rod

selected is invalid then the procedure displays a warning message and prompts for rod selection again by calling the first helper function.

- Finally, write a third helper procedure to have three input parameters: the array of rods, the player id, and the rod selected by this player. This procedure will validate whether the rod selected has at least one stone. If the rod selected is invalid then the procedure displays a warning message and prompts for rod selection again by calling the first helper function.

2) Using the following helper function, prompt and read how many stones to remove from the chosen rod. If the player inputs an invalid number of stones then display a warning message and prompt again. The number of stones to remove must be positive and must not exceed the number of stones on the chosen rod. The following helper function will be called from this procedure and will perform the following task described below. This procedure will check if the returned value from the helper function is valid and prompt the user again if necessary.

- Write a helper function to have two input parameters: the number of stones on the chosen rod and the rod id. The function will prompt the player for the number of stones to remove from the indicated rod and returns this value. Note this helper function will NOT perform any input validation checks. Instead the calling procedure will perform validation on the returned value from this helper function.

15. Write a procedure to implement step (vii) of the algorithm. Define the procedure to have three input parameters: the array of rods, the rod id of the chosen rod, and the number of stones to remove. This function will modify the array of rods by subtracting the specified number of stones from the chosen rod.

16. Write a procedure to implement step (ix) of the algorithm. Define the procedure to have a single input parameter that holds the player id. The procedure will print a message congratulating this winning player. The message should identify who won (player 1 or player 2).

17. Write a procedure to implement step (xiii) of the algorithm. Define this procedure to have a single input parameter that holds the player id. This procedure will switch the turn to the other player. In other words, if the player indicated in the input parameter is 1, then the procedure should change this value to 2. If the player indicated in the input parameter is 2, then the procedure should change this value to 1.

18. Be sure to modify the header comments "File", "Created by", "Creation Date", and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.

19. Be sure that there is a comment documenting each variable.

20. Be sure that your if statements, for and while loops and blocks are properly indented.
21. Check your output against the output from the solution executable provided.
22. **Test your code thorougly!**

4 Sample Program Interaction

Test your program against the program `pickstones_solution.exe`. The input and output of your program must match the input and output of the solution.

5 Program Submission

Submit your file `pickstones.cpp` in the hw5 drop box on Carmen. DO NOT submit the file `a.out`. If you do not submit your program, you will receive zero credit for the homework. If your program does not compile and run you will receive zero credit for the homework.