# CSCI 110 Intro to CS I

## Project 2

### Crafton Hills College

### Total Points: 50

#### Overview

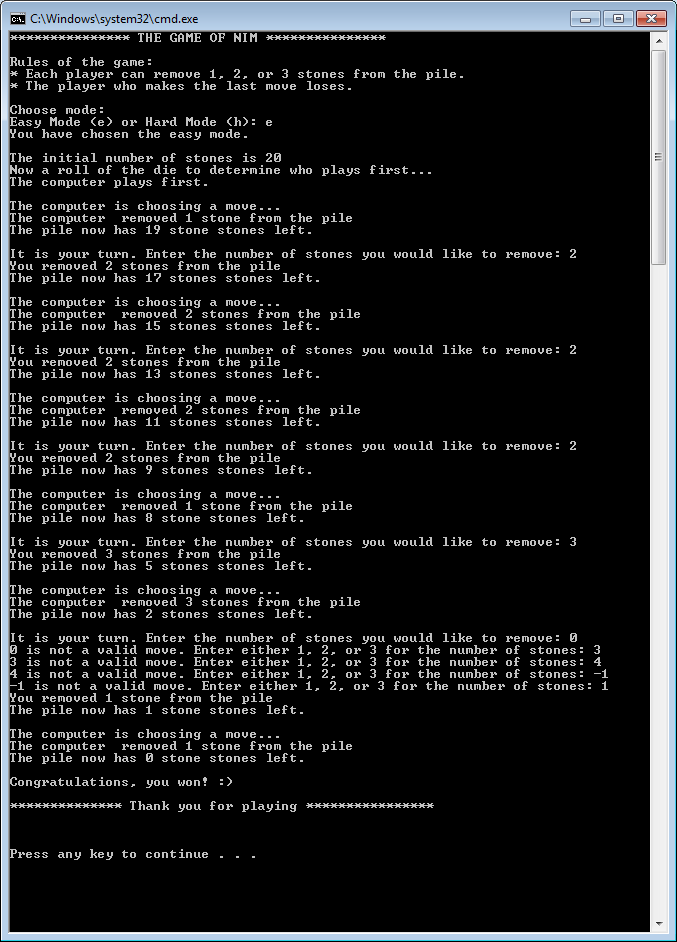
The game of Nim is a mathematical game of strategy with a number of variants. In this project, you will create a C++ program for a variant of Nim. Here are the rules of this game variant:

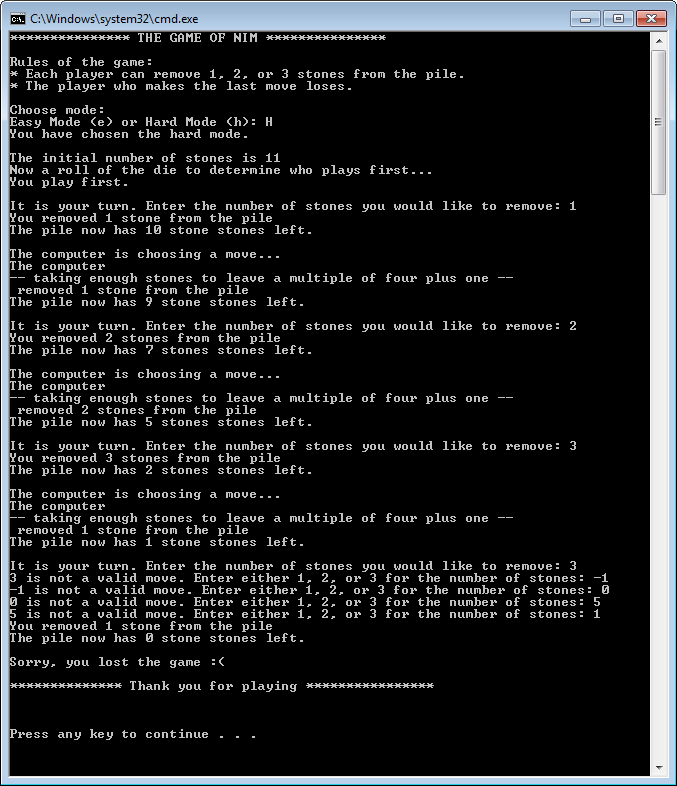
* Initially there is a pile of *n* stones, where *n* is a random integer between 10 and 21 (inclusive).
* Each player (the computer vs. the human player) alternates turns removing 1, 2, or 3 stones.
* The player who makes the last move loses.

The requirements of the program include:

1. The program starts by displaying an introduction of the game, including its rules.
2. It then asks the human player to choose between an easy mode and a hard mode.
3. The size of the pile of stones should be a random number between 10 and 21, inclusive.
4. The program chooses a player randomly to start the game (i.e. the computer and the human player should each have an equal chance of taking the first move).
5. When it is the human player’s turn, the human player should be allowed to remove 1, 2, or 3 stones but never more than the number of stones left.
6. When it is the computer’s turn, the behavior of the computer depends on the mode:
   1. *Easy mode:* The computer will pick a random number of stones (1, 2, or 3) to remove while never allowing the number of stones left to be negative.
   2. *Hard Mode:* The computer will always try to win when feasible.
      1. If the number of stones is a multiple of four plus 1 (e.g. 1, 5, 9, etc.), the computer has no winning strategy and will simply remove one stone.
      2. Otherwise, the computer will take enough stones to leave a multiple of four plus one. The computer can always win in this case. For example, if the number of stones in the pile is 8, the computer will take 3 stones to leave a pile of 5 stones (5 is 1 \* 4 + 1); regardless of the human player’s next move, the computer can always remove a number of stones to leave exactly one stone for the following turn by the human player.
7. The program should continue as long as the game is not over (a game is over when there is no more stones left).
8. The program should display outputs to clearly show the state and progress of the game, including:
   1. the initial number of stones
   2. the number of stones taken by the player and the resulting number of stones in each turn
   3. warning about invalid inputs, while allowing the user to reenter input when needed
   4. whether the human player wins or loses

See the following screenshots for sample outputs:





(Optional) If you would like to introduce time delay in your program (for example between the computer turn and the human turn in the game), do the following:

#include <Windows.h>

…

int main() {

…

Sleep(500); // delay for 500 milliseconds

…

}

#### What you need to submit for the project:

* A report containing the following items:
  + Title page, with your name
  + Pseudocode
  + Printout of source code
  + Sample screenshots running with representative inputs, such as the examples shown on the previous pages (use the [Alt+Prt Scr] buttons on your keyboard to capture a screenshot of your program)
    - Provide at least one for easy mode and one for hard mode
    - Include runs with invalid inputs
    - Always provide sufficient screenshots to demonstrate the correctness of your program.
  + A brief discussion of your project experience
    - Did you enjoy this project? What problems did you encounter?
    - What did you get out from the project?
    - How did you find the project (too easy, easy, just right, difficult, too difficult)?
    - What type of help/references did you use in your project (e.g. book, web sites, classmates, tutors)?

#### Grading Criteria

* Satisfaction of project requirements (40 points)
* Report (5 points)
* Coding style (5 points)
  + Provide comments about your program in appropriate places.
  + You must use the program template provided on Blackboard that has a program comment block containing filename, description, author, class, and date.
  + Use descriptive identifiers.
  + Define constants where appropriate.
  + Use proper spacing and indentation (refer to the textbook’s program style).