Project 1

<Battleship Game>

CSC-5 40651

Name: John Olmos

Date: 2/1/18

# Introduction

Battleship is a two player guessing game. The game is played on two square grids on which players place a number of ships. Players take turns picking elements within the grid to try and “hit” the opposing player’s ships until all elements have been hit and their fleet is destroyed.

# Game Rules

A typical game starts out with two 10x10 grids per player using numbers and letters to help identify elements within the grid. One grid is used to arrange the player’s ships and record their opponents picks while the second grid is used to record the player’s own picks.

Before the game starts each player arranges a ships in secret either vertically or horizontally within the grid. Each ship occupies a number of elements in the grid depending on its type making sure not to overlap with any other ship.

After each player has placed their ships the game starts in a series of rounds. During a round each player picks and element within the opponent’s grid in an attempt to hit one of their ships. Their opponent then announces if the pick was a “hit” or a “miss” on one of their ships and the result is recorded on a grid. If all of the elements of a ship have been hit it is considered sunk. The game ends once all ships have been hit and sunk.

# Development Summary

## V1

The first version of the game simply focused on getting the program to work. This meant scaling the game down to a single 1x10 row and limiting each side to one ship. The first obstacle was creating a way to allow the user to place a ship and then correctly updating a grid depending on their input for each round. Once this was done the process was mirrored and adjusted to allow the program to act as an opponent. Most time was spent on correctly updating the board and creating some basic form of logic for the computer’s turns.

## V2

With the program essentially functional, v2 was focused entirely on incorporating more of the missing elements of the checklist into the program. The most notable changes in this version were the addition of new libraries, new outputs, and incorporating external files.

## V3

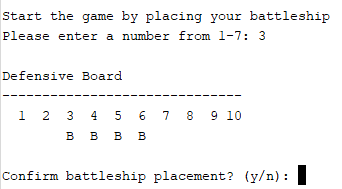
The third iteration of the program made no significant additions or removals but instead focused on making the code more uniform and reorganizing individual snippets of code within the program to match the turn sequence and output.

## V4

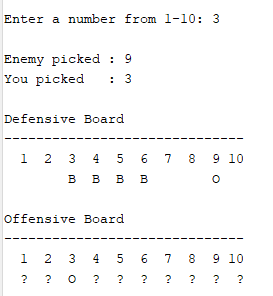
The last version of the program focused on including the remaining items on the checklist and improving the comments to help better explain the process of the program.

# Example Inputs/Outputs

The first prompt in the game asks users to place and confirm their ship placement along the row by entering a number [1,7].

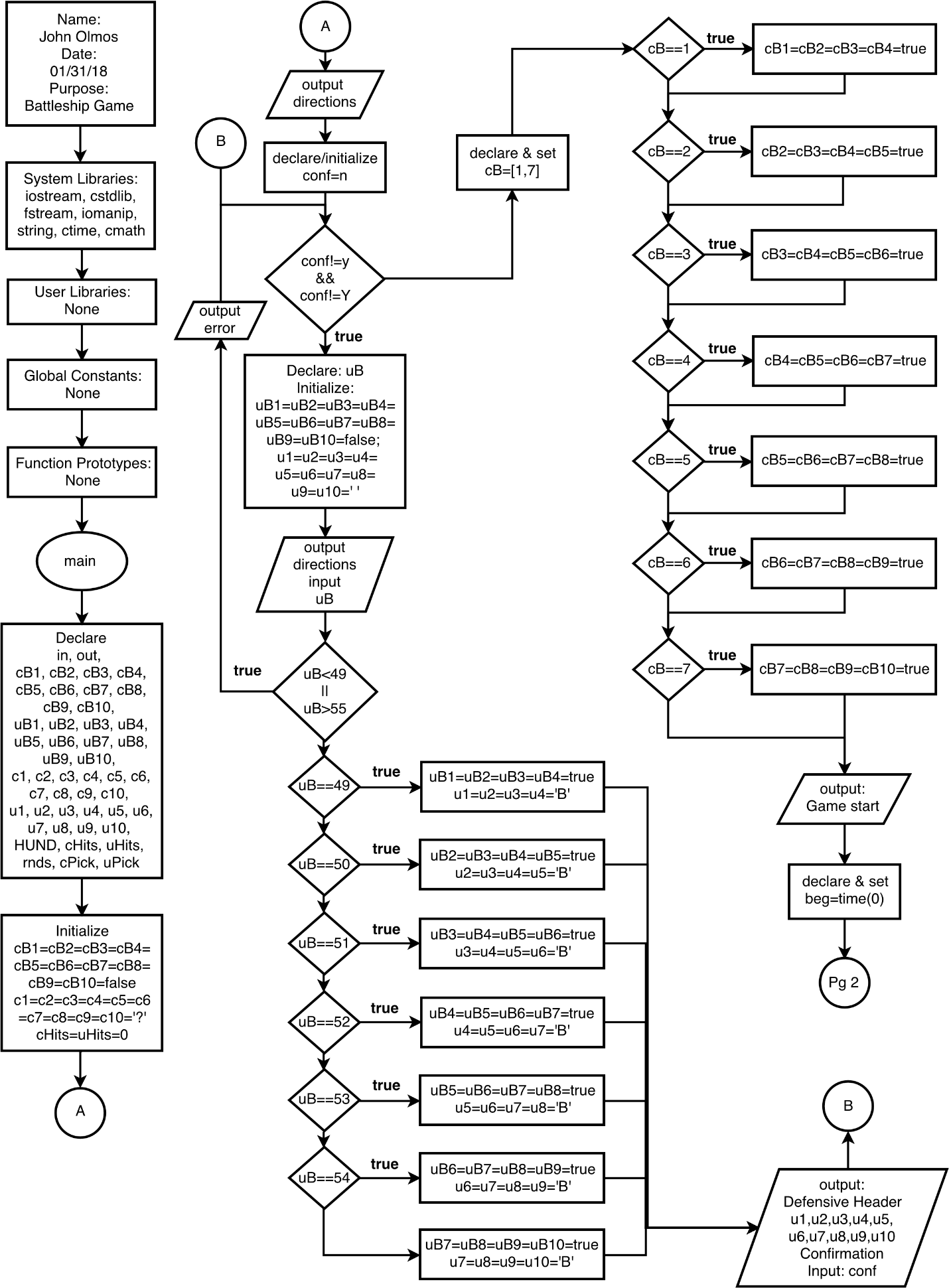


On the image to the right the program has output the player’s board after they entered the number 3 for their ship placement. Here the letter B is used to represent the ship.

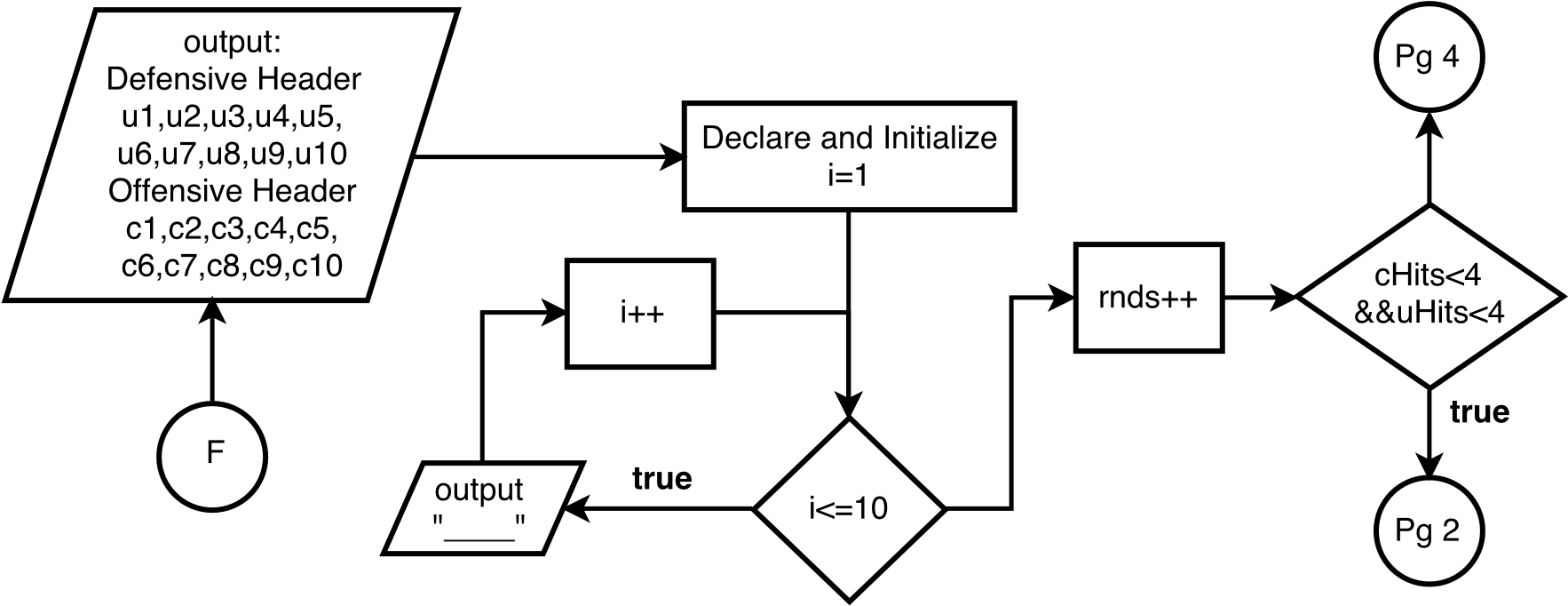
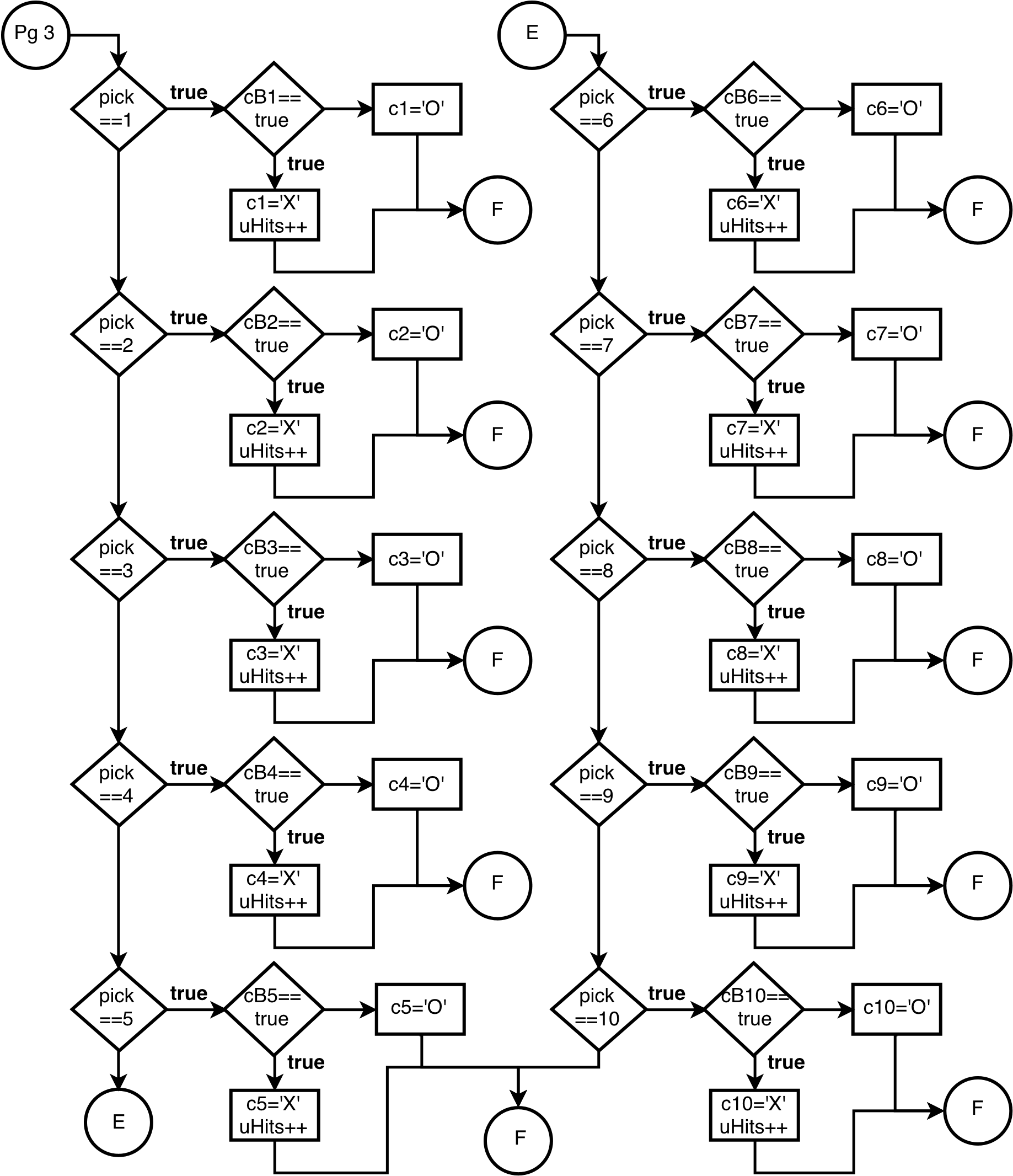


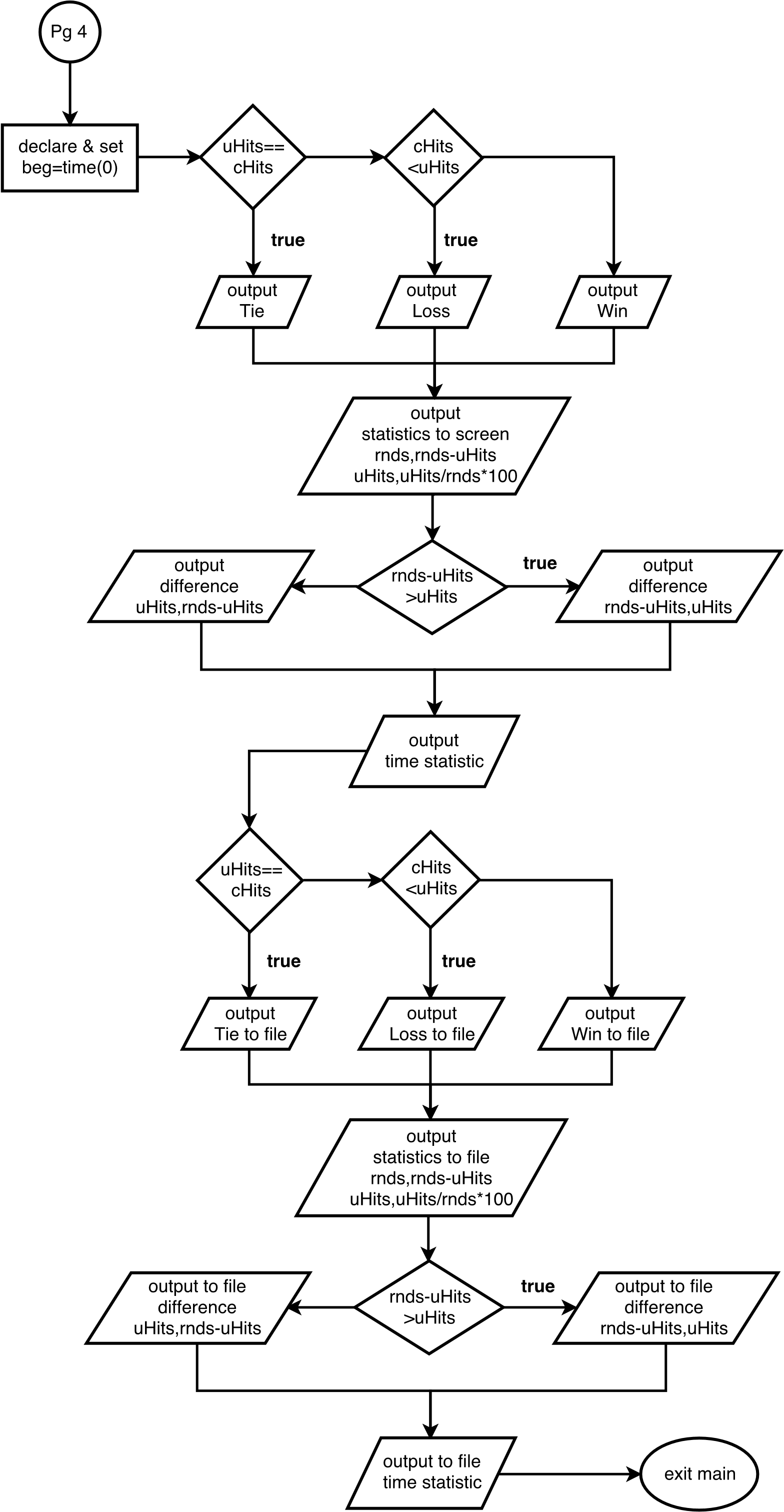
Later on the game asks users to enter a number [1,10] to try and hit the opponent’s ship. Once entered, the program outputs both the player and opponent choices along with the defensive and offensive boards.

If a pick manages to hit a ship, then the board outputs an X. However, if the pick misses the program marks the board with an O as seen on the image to the left.

Flowchart

# 





# Pseudo-Code

/\*

\* File: main.cpp

\* Author: John Olmos

\* Created on Jan 30, 2018, 12:06 AM

\* Purpose: Simple Battleship

\*/

//System Libraries Here

//Input - Output Library

//Srand to set the seed

//File I/O

//Format the output

//Strings

//Time library

//Math functions

//User Libraries Here

//Global Constants

//Such as PI, Vc, -> Math/Science values

//as well as conversions from system of units to another

//Percentage Conversion

//Function Prototypes Here

//Program Execution Begins Here

//Set the random number seed

//Instantiate and Open files

//Declare Variables

//I/O Files

//Computer battleship location

//User battleship location

//Offensive Board

//Defensive board

//Constant for 100

//User hit counter

//round counter

//Computer pick

//User pick

//Input or initialize values Here

//Open I/O files

//Last value in file becomes rounds

//Clear offensive ship

//Mask offensive board

//Reset hit counters

//Output instructions

//Set confirmation to 'n'

//Loop until ship is set and confirmed

//Clear defensive ship & board

//Output directions

//Input ship location & validate

//Set user battleship location

//Output Defensive Board

//Confirm

//Generate and battleship location & place

//Output Game start

//Play the game by -> Looping until user or computer has 4 hits

//Start time

//Computer turn -> Loop if element has been picked before

//Generate Computer pick

//Computer pick result

//For hits: change element to X and increment computer hit

//For misses: change element to O

//User turn

//Input user pick

//validate input

//User pick result

//For hits: change element to X and increment computer hit

//For misses: change element to O

//Output both picks

//Output Defensive Header & Board

//Output Offensive Header & Board

//Output divider and then increment round

//Stop time

//Output all the statistics to the screen

//Output all the statistics to a file

//Close files and Exit stage right!

# Check-off Sheet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Chap | Section | Topic | Where Line #''s | Pts | Notes |
| 2 | 2 | cout | 56-58, 66, 67, 72, 86-96, 116-121, 158, 164, 184-209, 217-225, 227, 228, 230, 231, 233 |  |  |
|  | 3 | libraries | 9-15 | 8 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals | 31-43 |  | No variables in global area, failed project! |
|  | 5 | Identifiers | 31-43 |  |  |
|  | 6 | Integers | 28, 39-42, 124, 169, 213 | 3 |  |
|  | 7 | Characters | 37, 38, 60, 62 | 3 |  |
|  | 8 | Strings | 43, 46 | 3 |  |
|  | 9 | Floats No Doubles | 225, 244 | 3 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 33-36 | 4 |  |
|  | 11 | Sizeof \*\*\*\*\* |  |  |  |
|  | 12 | Variables 7 characters or less | 31-43 |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables |  |  |  |
|  | 14 | Arithmetic operators | 222, 225, 226, 228, 231, 233, 241, 244, 245, 247, 250, 252 |  |  |
|  | 15 | Comments 20%+ | 1-256 | 5 | Model as pseudo code |
|  | 16 | Named Constants |  |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate |  |  | Emulate style in book/in class repository |
|  |  |  |  |  |  |
| 3 | 1 | cin | 68, 97, 159 |  |  |
|  | 2 | Math Expression | 222, 225, 226, 228, 231, 233, 241, 244, 245, 247, 250, 252 |  |  |
|  | 3 | Mixing data types \*\*\*\* |  |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* |  |  |  |
|  | 5 | Type Casting | 28, 225, 244 | 4 |  |
|  | 6 | Multiple assignment \*\*\*\*\* |  |  |  |
|  | 7 | Formatting output | 91-93, 194-196, 204-206, 224, 228, 231, 243, 247, 250 | 4 |  |
|  | 8 | Strings | 43, 46 | 3 |  |
|  | 9 | Math Library | 228, 231, 247, 250 | 4 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* |  |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators |  |  |  |
|  | 2 | if | 107-113 | 4 | Independent if |
|  | 4 | If-else | 71-98, 162-211, 226-232, 245-251 | 4 |  |
|  | 5 | Nesting | 77-97 | 4 |  |
|  | 6 | If-else-if | 77-83 | 4 |  |
|  | 7 | Flags \*\*\*\*\* |  |  |  |
|  | 8 | Logical operators | 61, 71, 133-151, 162, 163, 212 | 4 |  |
|  | 11 | Validating user input | 70-73, 161-165 | 4 |  |
|  | 13 | Conditional Operator | 133-152, 171-180, 217, 218, 236, 237 | 4 |  |
|  | 14 | Switch | 132-153, 170-181 | 4 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 171-180, 209, 210 | 4 |  |
|  | 2 | While | 61 | 4 |  |
|  | 5 | Do-while | 125, 127 | 4 |  |
|  | 6 | For loop | 209 | 4 |  |
|  | 11 | Files input/output both | 48-50, 236-252 | 8 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* |  |  | Failed Project if included |
| \*\*\*\*\*\* Not required to show | | | Total | 100 |  |