Program Design

## Input

1. **In detail indicate what the inputs to this program are?**

The input is the column number from 1 to 7 on which the user wants to drop their chip into.

1. **How do you validate the input?**

Only allow a number from 1 to 7 and check to see if the column is full or not.

1. **Pseudocode: Briefly explain how this section will look like**

Read the input in as an integer and use that input to determine which column of the board the chip will be place in. Check if that column is full or not and if it isn’t, allow the chip to be stored at the index.

## Processing

1. **What is this program?**

This program is a connect 4 game where a user will be playing again an AI but will also include an option for another player.

1. **Can we divide the calculations to the smaller “modules”?**

There will be many modules within this project. We will need a way to validate the board to check if a player has won or not, print the board each turn to allow to user to know what is going on, have a module to validate the user’s inputs and store the value into that index of the array. We will also need a module that will print out the whether the player has won or not and an entire function dedicated to the AI to make moves based on the user’s input and the board. It will incorporate algorithms to play against the user’s every move.

1. **Pseudocode: Briefly explain how this section will look like – Each module should have its own section.**

1: Ask for the user’s input from 1-7 and validate it.

2: Put the chip into the last row of the column and make sure each input will put the next chip on top instead of at the same index.

3: After the player’s turn the bot will go next and this loop will continue until a player makes a “connect 4”.

4: Read the board after each player’s turn to determine if they have won the game or not and if the win condition has been met, break from the loop and declare the winner.

5: Use a clear screen function throughout the program to erase previous boards as will continue to fill up the screen with outdated boards.

## Output

1. **What will this program display to the users?**

This program will display the board each turn after the player has picked a column.

1. **What kind of formatting this output will use (Currency with $ …)?**

Format the output to look like a connect 4 board with colors etc.

1. **Pseudocode: Briefly explain how this section will look like**

Use a loop to print out the board, which consists of a 2D array.

**FAQ**

1. **How does the scoring system work?**

The scoring system works by scoring each individual column and selecting the largest score. That score corresponds to the column and the AI will choose that column.

1. **How did you check the board to see if there was a win?**

I checked horizontals by counting the chips horizontally and resetting the count when there is an empty space or opponent chip in the cell.

1. **How does the clrscr function work?**

It clears the console using the window’s api header.

1. **Why did you not do a GUI?**

I did not plan to do a GUI nor did I plan to take an object oriented approach to this project. Transferring everything would take time that I do not have as of now.

1. **How smart is the AI?**

The AI can only see the present, or the board at its current state. Therefore, it cannot see into the future and a tactical plan may catch at AI off guard.

1. **Is this anyway of improving the AI?**

An algorithm called Minimax can be implemented to allow the AI to see up to n times into the future. This allows the AI to determine the best possible move making the AI unbeatable. The details of the minimax algorithm can be found on the Wikipedia page, but the main point is it is a search tree and the root node (first node) will be the maximizing player (BOT) and will try to get the highest score from its children. It uses recursion to go n times deep into the search tree. The larger the n the better the AI because n is how many times it will see all the board possibilities and all the choices the opponent could make. But since it is recursing so many times the AI’s time complexity will be increased. This is when alpha beta pruning can be added to decrease the amount of time it takes to run.

1. **Why are you using cin.clear and cin.ignore after your cin?**

This was added so if the user enters a character or string the program will not infinitely loop. It clears the error flags and ignores what is in the buffer allowing the user to enter another input without the program exploding.

1. **How does the center column score work?**

The center column scoring works by favoring the center by multiplying the score by a value which can be changed by the programmer. Since the center column is the most advantageous column the AI will favor it because it can create many possible wins through connecting a diagonal, vertical, or horizontal connect 4.

1. **Why is your AI using columns 1 to 7? Why is it not 0-6 since arrays start at 0?**

My DropChip function subtracts 1 from the column so It can be reused for other things. The AI calls this function therefore it must be from 1 to 7 otherwise there will be a bug.

1. **Why is best score initialized to -1000? Why is it a large negative number?**

Best score is initialized to -1000 because when it scores the window if the opponent is about to win it subtracts X value from the score (which results in a negative) and if the best score is not a large negative number there could be an error. For example, if best score were to be 0 it would be larger than the negative score even though the negative score should be favored as it is the largest score among the 7 columns.

1. **Why are you subtracting the score when the opponent has 3 and 1 empty space? Why not add a positive score?**

This is because it will subtract X value from the score and wherever the opponent is about to win that will be the largest number while all the other column will be negative values. Therefore, it will choose that column to drop the chip in.

1. **Why did you create a window? What is the window? Why are you scoring the window instead of the entire board?**

I put the chips into a window to have an easier time scoring them rather than looking at the entire board. It is easy to keep track of and less confusing.

1. **Why are you creating a copy of your board?**

This will allow the AI to drop their chip into all the spots and determine the best score and best col.

1. **What was the most difficult part of this project?**

I was encountering many bugs with the AI and counting the board, specifically diagonals as they are harder to input into code.

1. **What bugs did you encounter?**

One major bug that took many days of thinking of a solution was why sometimes the best score would be in a column that is full, therefore creating an infinite loop because the AI keep attempting to put their piece in that column and the program returning that the column is full but the AI still continues to put the chip in there.