1. Description of Your Project:
Project Synopsis: Our project is to design a checkers game that takes user input to select and move their desired pieces on the board in order to win using the command prompt.

The User/Player: The user will get to choose if they want to face off against another player on the board or against an AI. They will be prompted to choose the checker piece that they want to move and where to move it.

The AI: The AI will be using a random number generator in order to make a random move that is available to it. The AI could be improved so that it will always choose the option to jump over its opponent's piece if available.

The Checker Piece:
Every checker piece will be its own object of type Checker that is identified by a number. It will have the following functions: moves forward (move), checks if it can move(moveCheck), jumps over an opponent's piece (jump), checks if it can jump(jumpCheck), checks if it can jump multiple times in one turn (multipleJumpCheck), it gets promoted (promote), allows it to move backwards (promoteMove), checks if the piece has been jumped over(jumpedOver), removes the checker piece from the board if it has been jumped over(remove).

The Board: The board is an object of type Board that is a series of symbols that is printed out to the command console after every user input to showcase the current state of the board after a piece has been moved. It will check the positions of the checker pieces on the board. It will contain information on which space is occupied by whom or if it's empty. It will also check if a piece has reached the top or bottom rows of the board to promote it.

The Helper: It will tell the user when they choose a piece, the available spots it can move to or jump over. It's like an assistant.

Extra Features:
Create an AI object class. Actual visual of the board using a website or app. Score tracker. Turn timer.

2. Layered Development Schedule:

a. Functional minimum: (minimal items to make something that you might call an application. You'd be embarrassed if you only got this far, but at least it'd be something).

Create a player object class
Create checker piece object class
Create a board object class
void move() : Let the checker piece move forward.
void remove() : Removes the checker piece from the board.
void board(): Prints the board on the console.

b. Your low target: Your target for what you want to get done--the least possible to feel sort of OK about the result).

userInput(): Prompts user to input a checker piece and a position to move it to
bool promote() : Declares the piece as promoted to be able to move backward.
void Jump() : Makes the piece jump over the opponent's piece.

c. Your desirable target: (This is what you're aiming for if things go reasonably well).

bool moveCheck() : Checks whether or not a piece can move.
bool jumpCheck() Checks whether or not a piece can jump over another piece.
bool jumpedOver() : Called when a piece is jumped over.

d. Your high target: (It might be possible to get this much done if all goes extremely well)

void multipleJump(): Makes the piece jump over the opponent's piece multiple times.
bool multipleJumpCheck() : Checks if a piece can do consecutive jumps.
int PossibleMove() : Checks the possible moves a piece can make.
void promoteMove() : Same as move() but for backwards movement.

e. Your extras: (Stuff that you know you can't get done this semester, but you might add later if you decide to keep working on it after the class is over, just for fun).

Create an AI object class.
Actual visual of the board using a website or app.
Score tracker

3. Team Member's Role:
Zayd (Project Manager):
Layer 1: Create a player object class
Layer 2: void userInput()
Layer 3: bool jumpCheck()
Layer 4: void multipleJump() - bool multipleJumpCheck()

Noor (Tech Lead):
Layer 1: Create a board object class - void remove() - void board()
Layer 2: void Jump()
Layer 3: bool moveCheck()
Layer 4: void promoteMove()

Hasib (QA):
Layer 1: Create checker piece object class
Layer 2: bool promote()
Layer 3: bool jumpedOver()
Layer 4: int PossibleMove()

4. Assessment:
The main usage of the application will be by those who are interested in checkers. But we plan to make the user interface very beginner-friendly so anyone can use it. Users can learn the basic rules of checkers and put their newly learned knowledge into practice against another player or an AI opponent. As for the criteria used to judge whether or not this design is a success, we plan to play our application alongside a real checkers game program and see if that satisfies all the requirements of a proper checkers game from start to finish and use that as our basis for our standards. For example, the promotion of pieces or the act of eliminating an opponent's piece. These criteria should be met and able to be performed successfully for this design to be considered a success.