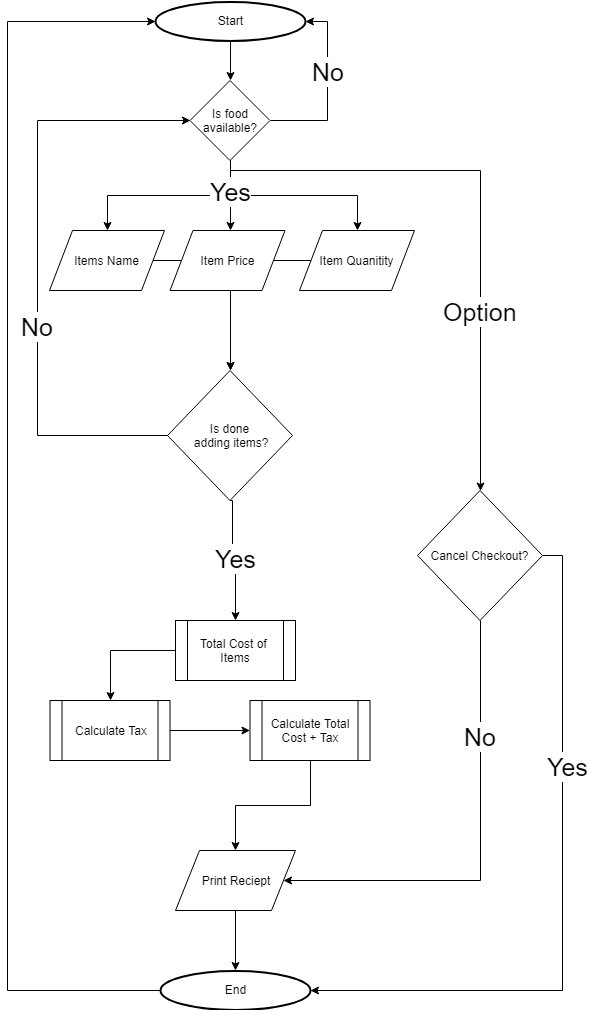
2-1 Short Paper: Pseudocode and Flowcharts  
Pseudocode: Painting a wall

START

GATHER wall height

GATHER wall width

CALCULATE wall area

CALCULATE number of gallons needed (area of wall / sqft per gal)

CALCULATE cans needed (area of wall // sqft per gal)

PRINT Wall Height

PRINT Wall Width

PRINT Wall Area

PRINT Cans needed

User PICKS Color of paint

PRINTS Color of Paint

PRINTS cost of certain color paint

CALCULATE Cans needed multiplied by Cost of paint color

PRINT Total cost of paint (cans \* cost of paint)Part Two

I find developing pseudocode easier but flowcharts are more intuitive. Pseudocode is easier since you do not have to plan for the flow of information but the information is “stacked” in a sense that you cannot use certain information unless declared in an above statement. With flowcharts you can visualize the flow of data much easier and see the processes involved in handling the data.

The challenges I encountered with flowcharts was adding more than needed to be added but you can add many more pathways in developing flowcharts and find out what features you need to add before beginning to code. Pseudocode is helpful in determining what objects you need upfront since you cannot use object that have not yet been previously determined in a statement above the statement called. The challenged with pseudocode is that you cannot go “sideways” and create dynamic pathways in the middle of your lift of instructions. For example, in the receipt flowchart I added a cancel option for the process to end the task early and start from the beginning. This would be confusing using pseudocode. If I were to add a cancel method it would be towards the beginning and forgotten or forgotten until the end and when developing the code, the developer would have a hard time seeing where the cancel feature would begin or end.

Developing programs without a blueprint is like developing a house without a plan. There is no foundation to what problem you are solving. Rooms are going to be mixed around like objects are going to be out of order from there they should be introduced into the program. Then the user or the person living in the house will not know what they should be looking at, what rooms to enter, and the house will be a maze. The programming process will be out of order and foundational programs that should be there to support other programs will be missing and the developer will be frustrated. I have tried to write programs without having a plan and everything halts when an object of function is missing. I have learned from previous experience how similar programming is to Legos. The engineer puts a few pieces together to create a complete piece. But that complete piece is just a small piece of a larger set. The blueprints are the foundation of all systems that engineers develop.