1090C Computer Programming I / 6090C Java Programming
Prof. Tom Wulf

Lab 1 – Flow Charts and Pseudo Code

10 pts

Labs are never about getting done… They are about learning.

I often will include a short mini-lecture at the start of the lab directions. This is designed to focus your attention on the learning goals for the lab and often provides you with specific technical details you will practice directly in the lab session.

Min

1090C Computer Programming I / 6090C Java Programming
Prof. Tom Wulf

Lab 1 – Flow Charts and Pseudo Code

10 pts

Labs are never about getting done… They are about learning.

I often will include a short mini-lecture at the start of the lab directions. This is designed to focus your attention on the learning goals for the lab and often provides you with specific technical details you will practice directly in the lab session.

Mini-lecture:

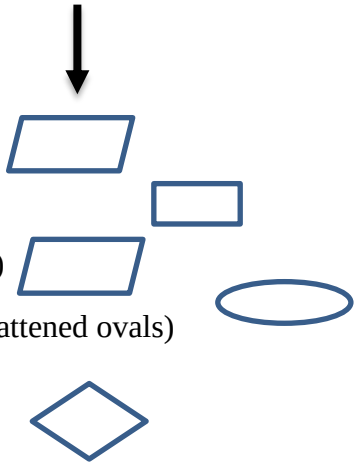
We use pseudo code and flow charts to capture the logic of a program prior to implementing it with code. In general practice, flow charts are too cumbersome for large programs/systems and so they are mainly used for isolated sub-routines, and high-level architectural patterns, etc.

We use pseudo code all the time. The pseudo code outline we write initially becomes the documentation for the completed program. To code the program, we go through and write the program code that corresponds to the pseudo code outline. The outline then becomes the program comments that document the code.

Students rarely believe me initially when I mention that the important part of this process is the generation of the pseudo code outline. Generating the code that corresponds to the outline is trivial and simply a matter of practice with the language syntax. If the pseudo code outline is incorrect then the program code will not correctly solve the task. Instead of trying to "write the program out of your head" as it were, take the time to create a solid and logically clear pseudo code outline before you begin coding.

Common flowchart symbols:

- Lines (Arrows) show the flow
- Input symbols (parallelograms)
- Processing symbols (rectangles)
- Output symbols (parallelograms)
- Terminal symbols (lozenges – flattened ovals)
- Decision symbols (diamonds)



i-lecture:

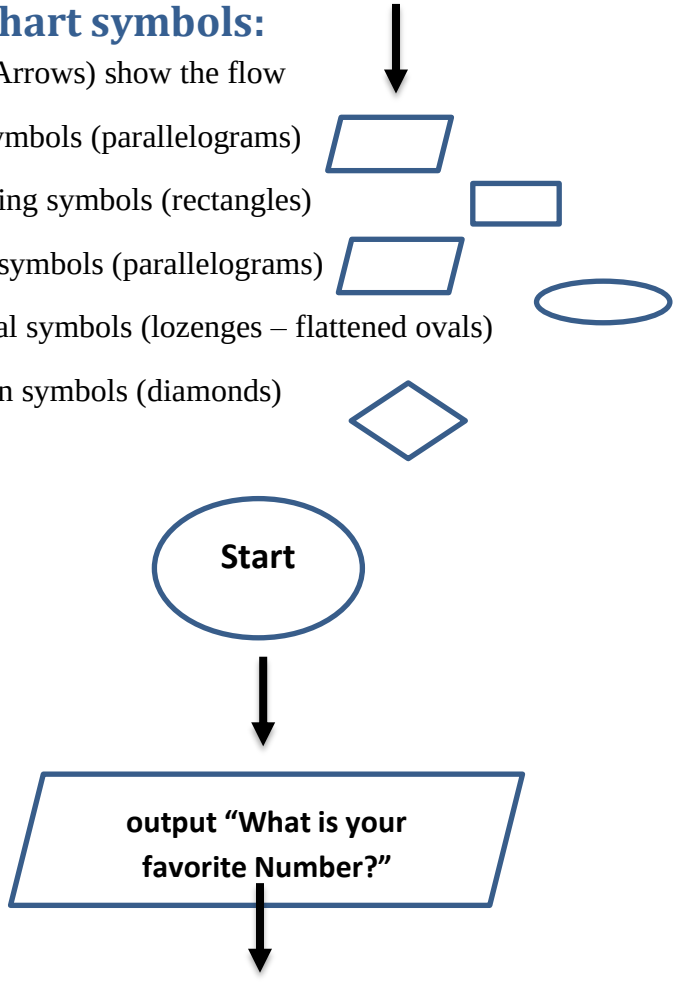
We use pseudo code and flow charts to capture the logic of a program prior to implementing it with code. In general practice, flow charts are too cumbersome for large programs/systems and so they are mainly used for isolated sub-routines, and high-level architectural patterns, etc.

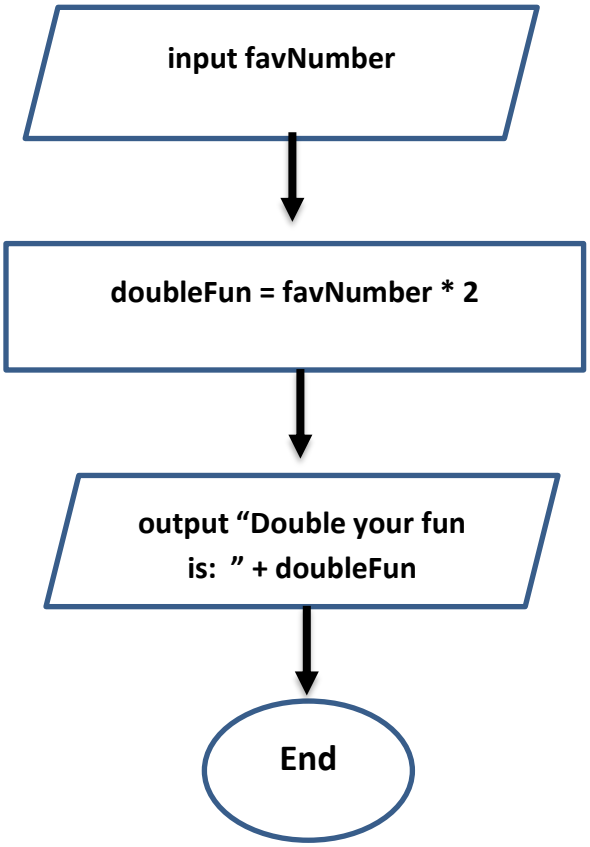
We use pseudo code all the time. The pseudo code outline we write initially becomes the documentation for the completed program. To code the program, we go through and write the program code that corresponds to the pseudo code outline. The outline then becomes the program comments that document the code.

Students rarely believe me initially when I mention that the important part of this process is the generation of the pseudo code outline. Generating the code that corresponds to the outline is trivial and simply a matter of practice with the language syntax. If the pseudo code outline is incorrect then the program code will not correctly solve the task. Instead of trying to "write the program out of your head" as it were, take the time to create a solid and logically clear pseudo code outline before you begin coding.

Common flowchart symbols:

- Lines (Arrows) show the flow
- Input symbols (parallelograms)
- Processing symbols (rectangles)
- Output symbols (parallelograms)
- Terminal symbols (lozenges – flattened ovals)
- Decision symbols (diamonds)





Pseudo Code:

Pseudo code is informal and fairly close to natural language. Rather than complete sentences, we tend to use very brief statements which will ultimately translate more directly to program code. Indentation is used to show groups of related statements, pay attention to it.

Look carefully at the example here and be sure to understand how both the pseudo code and the flowchart express the same logic (i.e. the same program).

Prompts: In every case, you have to **prompt** the user so they know what you need for them to input. Thus, before every input, we output the prompt msg so the user knows what we want to get from them.

Always include the prompt in your pseudo code and flowcharts when you do input.

Output should always be in the form of a complete sentence, not just a raw calculated value. i.e. *The total calculated building costs is: $560.00.*

Example:

```
start
    output "What is your favorite number?"
    input favNumber
    doubleFun = favNumber * 2
    ouput "Double your fun is " + doubleFun
stop
```

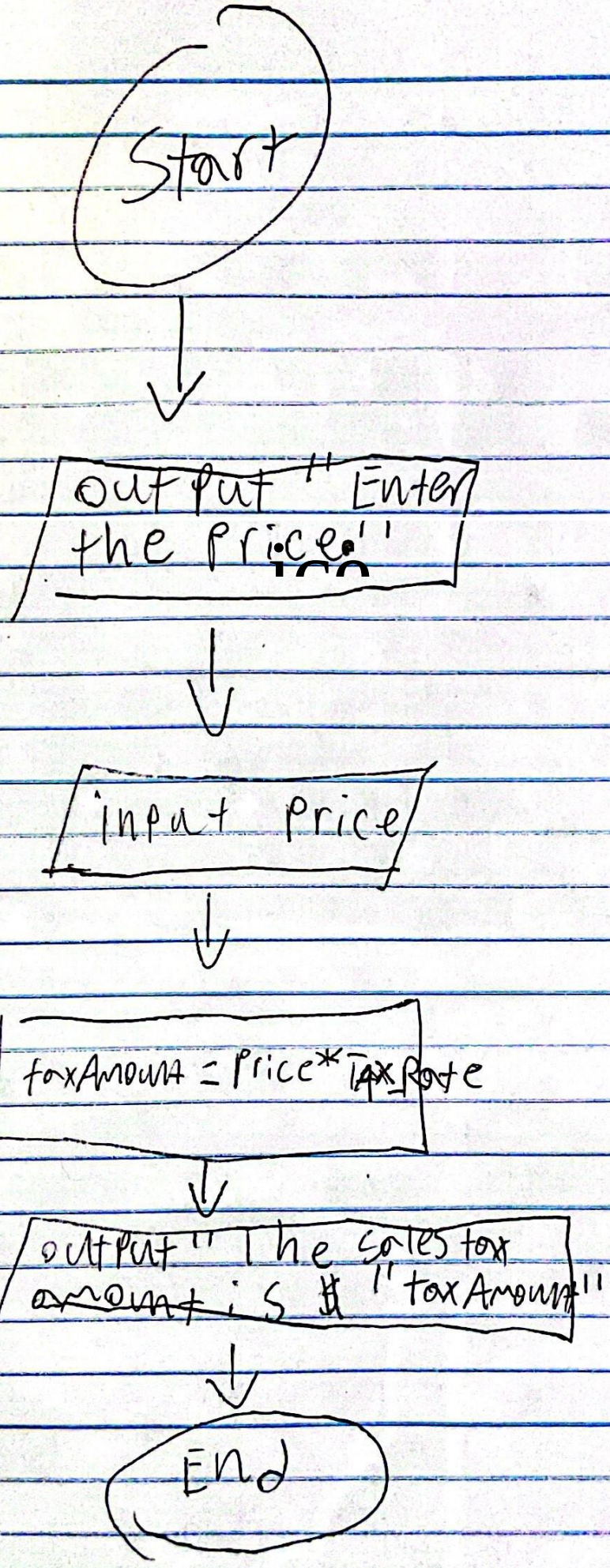
Lab:

1. Insert your work for the completed lab here within this MS Word document and submit it as directed at the end of the document.

2. For each of the following tasks, provide **both** the flow chart and pseudo code. You may use the MS Word symbols to create the flow charts or any other tool that lets you create and embed them in this document. Just insert your response right here in this document. Before doing the flow chart or pseudo code, list the input(s) and output(s) required for each program. (Please note that these are all simple programs with no branching so the flow charts will be a vertical connection of symbols like the example here.)

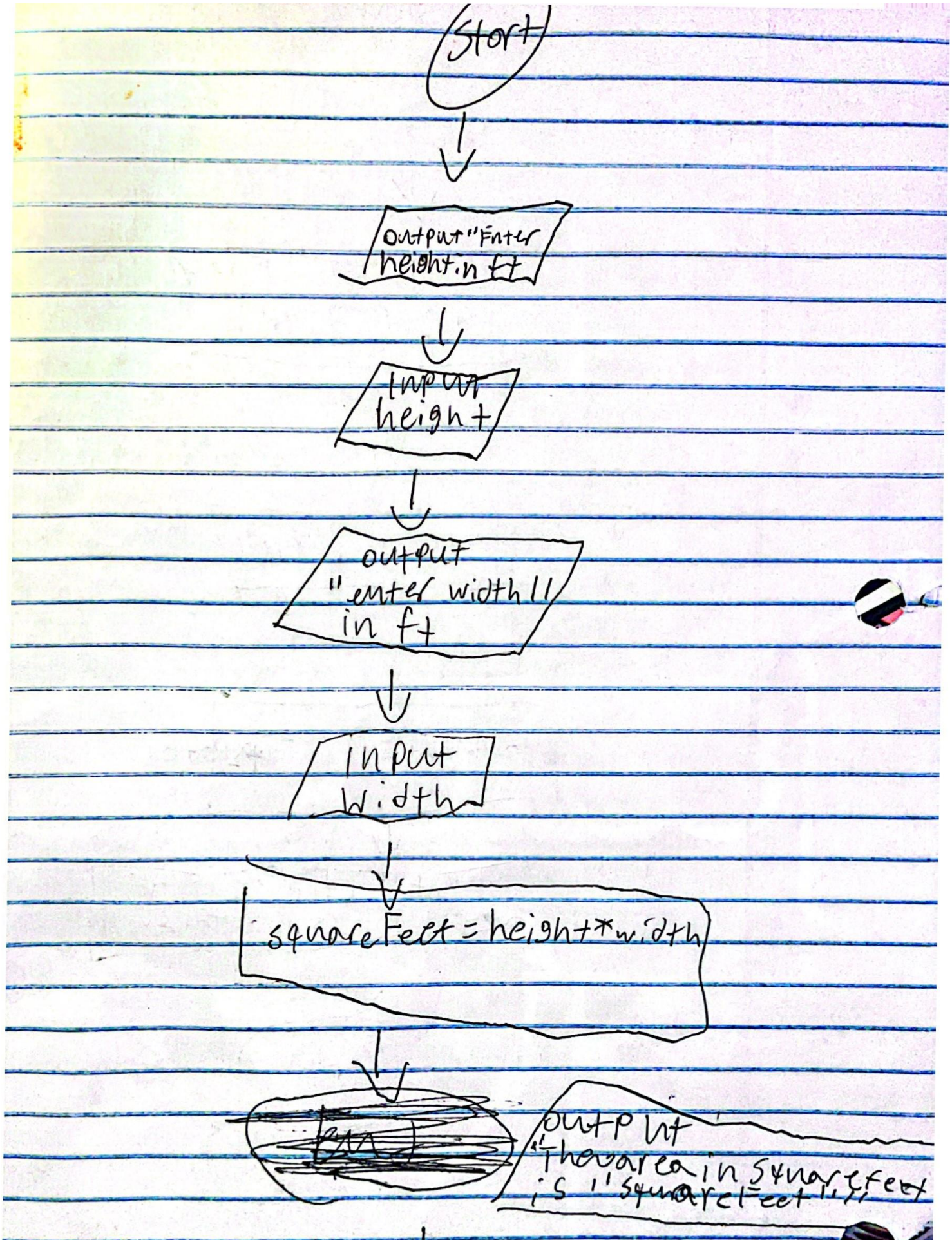
a. Task 1 (2 pts): a program where the user enters the price of a purchase and the program computes and outputs a 5% sales tax. Don't forget the prompt!

```
class TaxCalculator
  main()
   // Declarations
       num price
       num taxAmount
       num TAX_RATE = .05

     output "Enter the price."

     input price

     taxAmount = price * TAX_RATE

     output "The sales tax amount is $", taxAmount, "."

  return

endClass
```


Start
output "Enter the price"
input price
taxAmount = price * taxRate
output "The sales tax amount is $" taxAmount
End

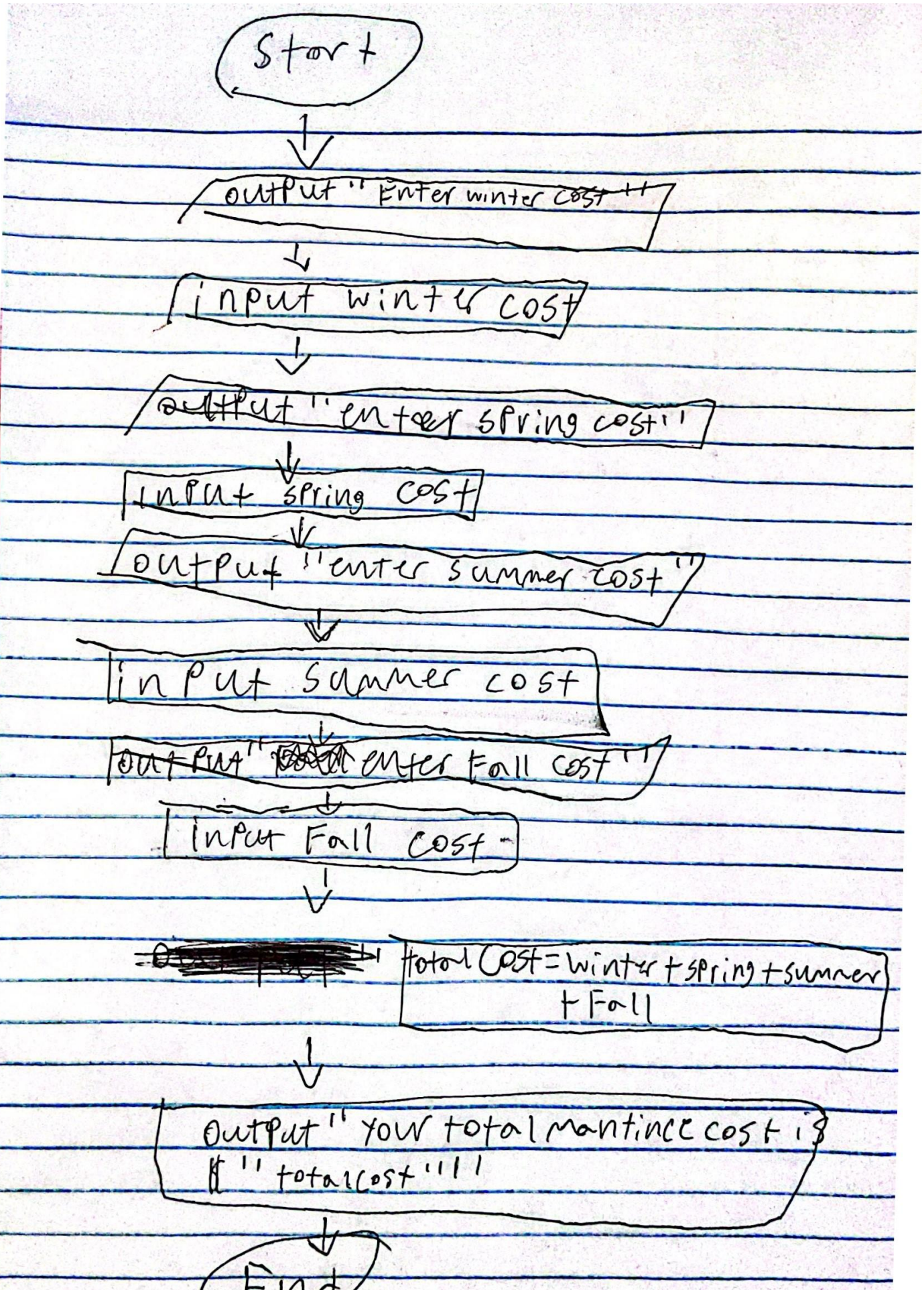
b. Task 2 (2 pts): A program for calculating the area in square feet. (User will input the height and the width in feet.)

```
class HeightWidth
  main()
     //conversion for height and width
       num height
       num width
       num squareFeet

       output “Enter the height in ft”

     input height

     output “ enter the width in ft”

     input width

     squareFeet = height*width

     output “The area in square feet is”,squareFeet,”.”

     return

endClass
```


Start
output "Enter height in ft
input height
output "enter width in ft
input width
squareFeet = height * width
output "The area in square feet is " squareFeet

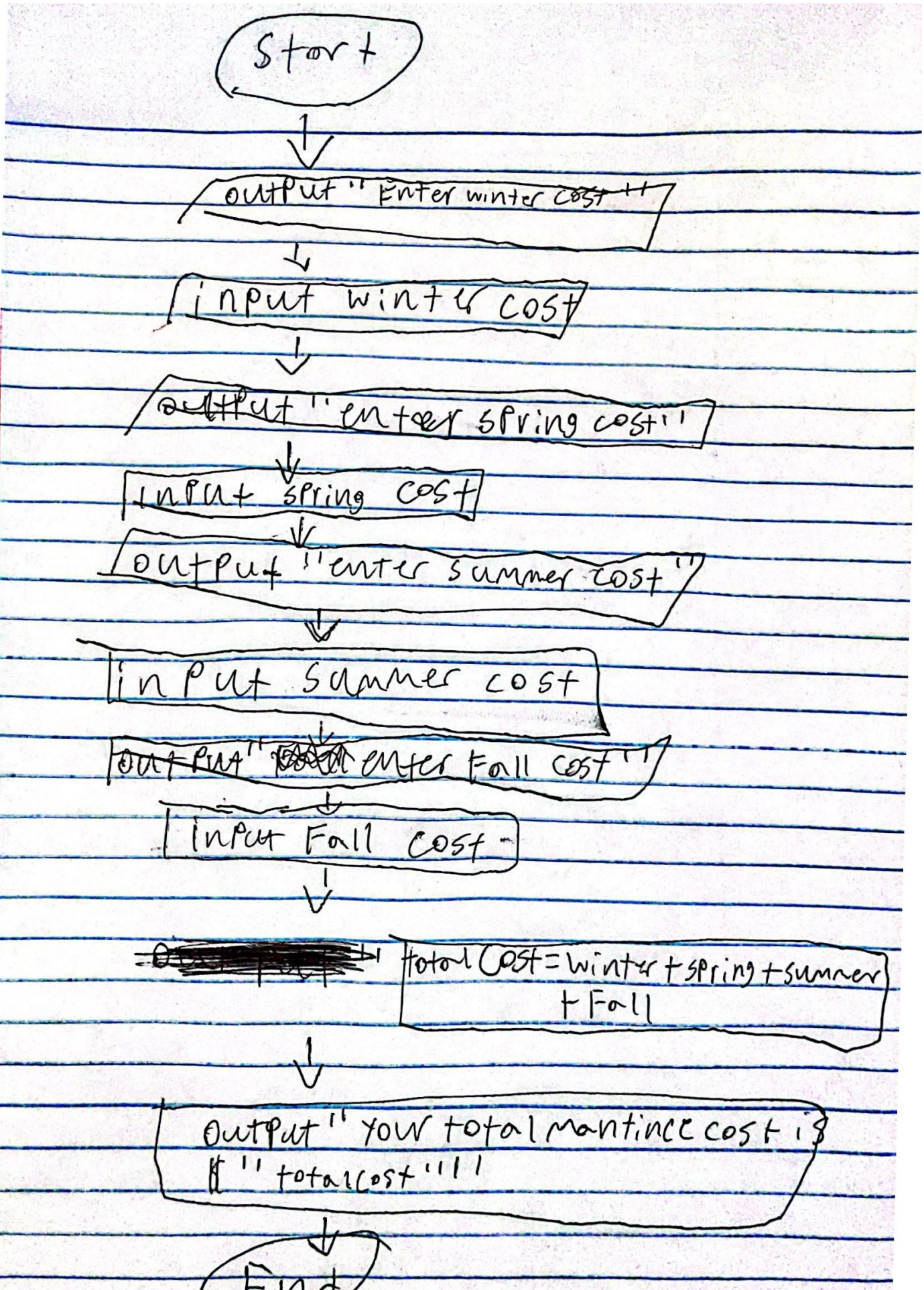
Task 3 (3 pts): A program that asks the user to enter home maintenance costs for each of the four seasons and returns the total yearly maintenance costs. (Separate prompt and input for each input value!)

```
class HomeMaintenance
    main()
    //declaration for the 4 seasons
    num winter
    num spring
    num summer
    num fall
    num totalCost

    output "please enter cost for winter"

    input price

    output "please enter cost for spring"

    input price

    output "please enter cost for summer"

    input price

    output "please enter the price for fall"

    input price

totalCost = winter+spring+summer+fall

output " The total cost for your home maintenance this year is
$",totalCost,"."
```


Start
output "Enter winter cost"
input winter cost
output "enter spring cost"
input spring cost
output "enter summer cost"
input summer cost
output "enter Fall cost"
input Fall cost
total Cost = winter + spring + summer + Fall
output "your total mantince cost is " "totalcost"
End

Task 4 (3 pts): A program that calculates the difference between two numbers. i.e. subtracts one from the other. Thus this might be a negative value.

```
class NumberDifferenceCalculator
  main()
    // Declarations
    num numOne
    num numTwo
    num difference

    output "Enter the first number:"
    input numOne

    output "Enter the second number:"
    input numTwo

    difference = numOne - numTwo

   output "The difference between", numOne, "and", numTwo, "is",
difference

  return
endClass
```


Start
output "Enter winter cost"
input winter cost
output "enter spring cost"
input spring cost
output "enter summer cost"
input summer cost
output "enter Fall cost"
input Fall cost
total Cost = winter + spring + summer + Fall
output "your total mantince cost is " "totalcost"
End

3. Submitting your work: carefully check your work.

   You need to follow directions and develop professional work habits. I'm very specific about how I want the work submitted especially with the naming and the format of files so they can be tracked efficiently. If you can't or won't follow directions, you work will not be graded and you receive no credit.

   Rename your copy of this word file as **Lastname_Firstname_Lab01.docx** using your name. Submit this file using the Canvas assignment mechanism.