Please Do Not Begin the Test Until Told to Start.

Do Not Detach This Page!

This test has eight questions worth a total value of 95 points. Note that some questions are worth more than others. Be sure to read over all of the questions carefully before answering any of them. A little thought can help you avoid wasting time and making careless errors.

This is a closed book, individual exam with one sheet of notes allowed. There is no sharing of materials, and no looking at neighbor's papers! Electronic aides (e.g. including but not limited to calculators, computers, sending/receiving devices) are not allowed. As stated in the syllabus, cheating on a midterm or final can result in course failure.

If you use additional pages for your answers, put your name on them, and fold them into this test.

Name
_____________________________________________________________________________
Please do not write below this line.

1. ______/ 20
2. ______/ 30
3. ______/ 5
4. ______/ 5
5. ______/ 5
6. ______/ 15
7. ______/ 5
8. ______/ 10

Total ______/95 (55 minutes)

Grading policies:
* Spelling and grammar, including completeness of sentences, COUNT in questions that ask for AN EXPLANATION. In all cases, care in following the directions in the question COUNTs.
* Incomprehensible answers get zero points! Inefficient or poorly expressed code, and answers that do not display understanding of course content might not earn full or even any credit, even if logically right.
1. (20 points) Give two separate sequences of statements. (a) The first will cause the robot to move forward and then rotate towards the right and then stop. (b) The second will cause the robot to move along a curved path that curves to the right, and then stop. The exact distances and angles don’t matter as long as they are non-zero.

(a) forward(1,1) turnRight(1,1)
# -1 for unnecessary operations.
# 2 pts for each right direction.
---an alternative----
forward(1,1) # full, 1 second
motors(1,-1) # start rotating right
wait(1)
stop()

(b) motors(1.0, 0.2) # The left positive value must be greater than a right positive value [4]
wait(1.0) # or other perceptible time (> 0.2 sec) [3]
stop() # [3]
---------an alternative-----------------
move (1.0, -0.2) # positive forward power, negative rotation.
wait(1.0)
stop()
# It’s wasteful of Bluetooth bandwidth to put motors or move
# operations in a while timeRemaining(..): loop (-1).

2. (25+5 points) Define a main() function that will ask the user for a value. Define another function work(x). Make main() (a) call work with the value and (b) print the string that work should return. The work function should first test to see if the value is between -30 and 120 (inclusive). If it is not, the function should return the string “You must not live on Earth”. If the value given is in that range, it should return the string “Very Hot” if the value is 100 or greater, if not, “Hot” if the value is 85 or greater, if not, “Nice” if the value is 75 or greater, if not, “Cold” if the value is 35 or greater, if not, and “Very Cold” if the value is less than 35.

The 5 additional points will be awarded if your program uses a loop that scans following list to find which string to return:

LL = [[100, 'Very Hot'], [85, 'Hot'], [75, 'Nice'], [35, 'Cold'], [-30, 'Very Cold']]

An important lesson in thinking like a computer scientist or serious programmer is that specifications for what a program must do, known as "requirements," whether you invent them yourself or you are given them by a customer, must be READ and FOLLOWED literally. That means the program must do exactly what the requirements say, NOT something like it or something easier or more familiar (or available in your notes!) merely suggested by the words of the problem! It is helpful to identify a set of "function points" from the specifications. Function point analysis of specifications is one of many software engineering techniques.

I will do that and use the function points as the basis for grading. Below is an exact copy of the question with the function points demarcated by separate lines. Since the distinction between a function printing something and a function returning something is conceptually deeper and more critical to sophisticated thinking than other details, it counts for more points in this exam.
Define a main() function (1 point)
that will ask the user for a value. (1 point)
Define another function work(x). (1 point for def., 1 point for one parameter.)
Make main() (a) call work (1 point) with the value (2 points)
and (b) print the string that work should return. (2 points)
The work function should first test to see if the value is between -30 and 120
(inclusive). (1 point for attempt, 1 for correctness.)
If it is not, the function should return (3 points)

The 5 additional points will be awarded if your program uses a loop
that scans following list to find which string to return:(2 points for
selecting the strings from the list instead of retyping them into your
code, 3 points for a correct loop that scans the list to find the right
string using an algorithm similar or equivalent to the engagement
ring shopping programs in the in-class exercises.)

```python
# Other solutions are valid.

def main():
    iV = input('Value?: ')
    print(work(iV))

def work(x):
    if x < -30 or x > 120:
        return 'You must not live on Earth'
    if x >= 100:
        return 'Very Hot'
    if x >= 85:
        return 'Hot'
    if x >= 75:
        return 'Nice'
    if x >= 35:
        return 'Cold'
    return 'Very Cold'
```

```python
# Other solutions are valid.

def main():
    iV = input('Value?: ')
    print(work(iV))

def work(x):
    if x < -30 or x > 120:
        return 'You must not live on Earth'
    for L in LL:
        if L[0] <= x:
            return L[1]

# Other solutions are valid.

# A short alternative for main() is
def main():
    print(input('Value?: '))
```
3. (5 points) Computer scientists who invent algorithms for compiling or interpreting languages like Python need to be clear about where each statement begins and ends. In the following code fragment, encircle the ENTIRE if statement:

```python
exp = input('Should robot explore or just back up beep? 1 or 0: ') for i in range(0, 10):
    if exp == 1:
        print 'robot is beeping'
        beep( 1, 1000 ) # 1 second duration, 1000 hz. frequency
        while getStall() != 1:
            forward( 0.7, 1 ) # 0.7 power, 1 second
            turnRight( 1.0, 0.5 ) # full power, 0.5 second
        backward( 1, 0.5 )
```

The circle must omit `backward( 1, 0.5 )`. Some people detected that (a) I missed the colon (":") in the `for` statement and/or (b) the `while` statement is indented one step too much. 1 bonus point for detecting each of those!

4. (5 points) Show what is printed out by the following python statements:

```python
x = 4
y = 3
z = 2
w = x + y * z
print w
```

5. (5 points) Show what is printed out by the following python statements:

```python
x = 8
while x > 0:
    print x
    if x > 5:
        x = x - 2
    else:
        x = x - 1
```

6. (15 points) Explain the difference between the assignment operation (=) and the equality predicate (==) in the following terms.

- (9 points) What do they do? Explain for = and for ==. Use 1-2 complete sentences for each.

The assignment operator makes the computer evaluate the expression on its right side and then assign the resulting value as the value of the variable on its left side.

The equality predicate makes the computer evaluate the two expressions on each side of it and the determine whether the two resulting values are equal. It then returns True or False depending on whether or not those values are equal, so that the computer can make a decision in an if or a while statement accordingly.

- (4 points) What kind of Python code can be on the left-hand side of = and what can be on the left-hand side of ==? What can be on the right-hand side of = and what can be on the right-hand side of ==?
Left side __ a variable __ = Right side __ an expression __

Left side __ an expression __ == Right side __ an expression __
"expression" was worth 3 points, one or more forms of expressions ("integer", "float", "constant", "variable", etc.) were worth 2 points, just examples (NOT kinds) were worth only 1 point, and a point was deducted for not writing the same thing for BOTH SIDES of the equality testing operator (==).

● (2 points) What kind (type) of value does each return, if any?

Type of the return value of == __ bool or Boolean or boolean or True/False __
Type of the return value of = __________________________
(Explain if you cannot answer either or both questions.)

The second question cannot be answered because in Python, the assignment operator DOES NOT RETURN ANY VALUE. This is in contrast to C, C++ and Java where the assignment operator returns the value that is assigned. (I don't know but can look up the rules when a type conversion occurs.) When you try something like `print (x = 3)` or `print x = 3` in Python, a syntax error is detected and reported. (Try it!)

This is a trick question and an answer to the effect it returns that the value that is assigned is a good guess, but wrong. Only a couple of students got it.

`bool` is the Python token that stands for the Boolean type.

7. (5 points) WRITE on the image of the debugger below exactly how the Locals information changes after the "Step" button is pushed once.

Note that the whole program is NOT given. The ONLY right answer is to cross out the 13 and write 41 next to Z. -2 for bad arithmetic. 0 if the other variables are changed, you indicated they become undefined or just explained how they changed. This question tests whether you actually practiced using the debugger as assigned and whether you understand variables, values and how assignment statements calculate values, use variables, and change values of variables.
8. def runDemos():
    while True:
        print 'A Boring Line'
        input('Type your choice,0,1 or 2:')
        CH = 1 #I put this in to remove that
        #bloody red NameError message.
        if CH == 0:
            return
        elif CH == 1:
            demo1()
        elif CH == 2:
            demo2()
        else
            print 'Baahhh!'
(10 points) Norbert W. Simpliticus' program originally did not have the line with the comment that
begins "CH = 1." However, when he ran the program by calling runDemos(), there was a NameError.
He then added that line so that the name error didn't occur and the program seemed to work OK as far
as he tested it.
(a. 5 points) Explain what is now wrong with this program, which is supposed to enable the user to
choose and rechoose among two different demos or to finish. (Complete sentence, not like this.)

The value of CH is used to select the choice of which demo or to finish, but instead of getting
its value from user input, the program disregards the value inputted by the user and assigns CH the
fixed value 1. The result is that the program calls demo1() after every user input, regardless of what
number the user gave.

(b. 3 points) Simpliticus didn't test it enough. What input(s) demonstrate it fails? Completeness counts!

For the 3 points, three test inputs should be mentioned: 0, 2, and some number
different from 0, 1 and 2. I gave full credit for an answer equivalent to "any other than 1."
but I wrote that it is best in situations like this to figure out the ways a program can fail and a
different test input for each way. In software engineering, the degree to which all possible
execution paths in a program are tested by different test cases is called the "coverage" of the
testing.

(c. 2 points) Fix it! (Correct it by crossing out and/or writing on the code.)

One person surprised me by crossing out the print statement. I like that because the I tried to be funny but
instead I made the print statement useless for the purpose of the program. Anothers got a bonus point for
detecting the missing : in the else clause.

A right answer must include:
    CH = input('Type your choice,0,1 or 2:')
    cross out CH = 1