Name

**Computer Science 220**

**Test 3**

**November 21, 2013**

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Points | Your Points | Course Outcomes |
| 1 | 6 |  | 8, 9 |
| 2 | 10 |  | 8, 9, 15 |
| 3 | 10 |  | 2, 7, 10, 15 |
| 4 | 10 |  | 2, 8, 9, 10 |
| 5 | 10 |  | 2, 8, 9, 11, 15 |
| 6 | 8 |  | 2, 8, 9, 11 |
| 7 | 8 |  | 14 |
| 8 | 6 |  | 13 |
| 9 | 6 |  | 13 |
| 10 | 4 |  | 4, 12, 15 |
| Bonus | 2 | 2 |  |
| Total | 80 |  |  |

* Write your name ONLY on this cover page.
* Turn off cell phones, calculators and anything that makes a noise or can be used for calculations.
* Nothing is allowed on the desk top but the test, pens, pencils, erasers, and drinks.
* Do not ask a question in a way that gives away the answer to the test question.
* In any code that you write on this test, you do not have to write comments or print statements that explain what the program does. Just write enough code to do what is asked.
* If the code requires import statements, you must include them.

1. Apply deMorgan’s law to the following:
2. not(x <=5) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. not(y != 7) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. not (x >9 or y >= 8) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. not(x == 10 and z < 5) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. Write a function isEqual(value1, value2) 🡪 Boolean that accepts two calculated float values and returns True if these values are equal and False if they are not.
7. Write the body of the function to help a doctor diagnose a patient’s blood pressure, bloodPressure(systolic, diastolic) 🡪 String. This function accepts two numbers related to blood pressure and returns the appropriate string (“Normal”, “Prehyper”, “High”) based on the chart below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Systolic** |  | **Diastolic** | **Result** |
| Less than 120 | and | Less than 80 | Normal |
| 120-139 | or | 80-89 | Prehyper |
| 140 or higher | or | 90 or higher | High |

def bloodPressure(systolic, diastolic):

1. Write a code segment that asks a user to create a password and decides whether or not the password is legitimate. The code should run until the user enters an acceptable password. Once a good password has been entered, the password should be output to the user. The rules for a password are:
2. The last character must be a ‘!’ or the first two characters must be ‘ab’
3. The password must be at least 6 characters long.
4. Myrtle the Mathematician has downloaded a version of Python in which the list index() method does not work properly. Write an efficient method for her, index(items, value), which looks through a list, items, for a particular item, value. The method should return -1 if value is not found in items. If value is found in items it should return the position of the first occurrence of value in items.
5. Trace the values of the variables in the following code segments.

a.

i = 10

total = 0

while total < 25 or i >= 7:

i = i – 1

total = total + i

b.

i = 0

total = 1

while not(i < 5 and total >= 16):

total = total \* 2

i = i + 1

1. Assume that the following list has been created in memory. Trace is contents as they would change if the list was sorted using a selection sort.

values = [100, 35, 12, 27, 75, 40, 50]

1. Linear search complexity:
   1. Assume that a list of 5000 ***unsorted*** items will be searched using an efficient linear search, in the worst case approximately how many comparisons will be made? Briefly explain your answer.
   2. In the best case, how many comparisons will be made? Briefly explain your answer.
2. Binary search complexity:
   1. Assume that a list of 5000 ***sorted*** items will be searched using a binary search, in the worst case approximately how many comparisons will be made? Briefly explain your answer.
   2. Assume that a list of 5000 ***unsorted*** items will be searched using a binary search, in the worst case approximately how many comparisons will be made in order to *accurately* perform the binary search? Briefly explain your answer.
3. What is output by the following code?

def alter1(values):

values.sort()

values.reverse()

def alter2(values):

newValues = []

for value in newValues:

newValues.append(value)

newValues.sort()

newValues.reverse()

def main():

nums = [5, 3, 7]

alter1(nums)

print(nums)

nums = [5, 3, 7]

alter2(nums)

print(nums)