Note for 2010 Students:
Anything with a gray background covers a topic that is not applicable to the Fall 2010 Final exam. Print statements have been updated to Python 3.1 syntax.

HAND IN
Answers Are Recorded on Question Paper

QUEEN'S UNIVERSITY
SCHOOL OF COMPUTING

SOLUTION

CISC101, FALL TERM, 2008
ELEMENTS OF COMPUTING SCIENCE I
FINAL EXAMINATION
12 December 2008, 9am to Noon, Jeffrey Hall

Instructor: Alan McLeod

Please write your answers in the boxes provided. The back of any page can be used for rough work. This exam is 3 hours in length. Please put your student number at the top of each page. Extra space is provided on the last page of the exam

This is a closed book exam. No computers or calculators are allowed or even needed.

PLEASE NOTE: "Proctors are unable to respond to queries about the interpretation of exam questions. Do your best to answer exam questions as written."

Student Number:

1.	/ 10	5.	/ 10
2.	/ 6	6.	/ 10
3.	/ 10	7.	/ 18
4.	/ 4	8.	/ 12

TOTAL: / 80

Problem 1) [10 marks]

Write the output of the following complete program beside each print statement.

```
def main() :
    print(-1 * 4 + 3 / 4)                    -4
    print(4 ** 2 / 4)                        4
    print(15 % 4)                            3
    print((7 + 2) * (3 + (6 - 4)))           45
    print (19 // 20 * 20)                    0
    print(int(4.9))                          4
    print "%.1f cm per inch" % 2.54          2.5 cm per inch
    print("Hi" * 3)                          HiHiHi
    print("Ho" + "Ho" + "Ho")                HoHoHo
    print("Silver\n!")                       Silver
                                             !
main()
```

Problem 2) [6 marks]

What is the value of the base 2 number: 111100001111 in base 16 (hexadecimal)? **F0F**

What is the value of the base 10 number: 47 in base 2 (binary)? **101111**

What is the value of the base 2 number 101.11 in base 10? **5.75**

Work area:

Problem 3) [10 marks]

Write the output of the following complete program beside each print statement.

```
def main() :
    print([1, 2] * 3)                          [1, 2, 1, 2, 1, 2]
    print([4, 5, 6] + [1, 2, 3])               [4, 5, 6, 1, 2, 3]
    print(7 not in [3, 4, 5, 6, 7, 8])         False
    print(len((4, 5, 6, 7)))                   4
    print([4, 5, 6][0])                        4
    print(range(5))                            [0, 1, 2, 3, 4]
    print(range(2, 10, 2))                     [2, 4, 6, 8]
    aList = [7, 8, 9, 10, 11]
    print(aList[2 : 4])                        [9, 10]
    print(aList[-1])                           11
    print(5 in aList)                          False

main()
```

Problem 4) [4 marks]

Indicate the list generated by each of the following list comprehension statements:

```
[2 * (x % 5) for x in range(5, 11)]
```

[0, 2, 4, 6, 8, 0]

```
[2 * x for x in range(10) if x > 4]
```

[10, 12, 14, 16, 18]

Problem 5) [10 marks]

Write the output of the following complete program beside each print statement.

```
def aFunction(aNum, aList1, aList2, aList3) :

    aNum = 0
    for i in range(len(aList1)) :
        aNum = aNum + aList1[i]
        aList1[i] = aList1[i] * 2

    aList2.sort()

    aList3 = [4, 6, 8]

    return aNum

def main() :

    aNum1 = 5

    aList1 = [1, 2, 3, 4]

    aList2 = [5, 4, 2, 1]

    aList3 = [10, 12, 14]

    aNum2 = aFunction(aNum1, aList1, aList2, aList3)
```

`print(aNum1)`	**5**
`print(aNum2)`	**10**
`print(aList1)`	**[2, 4, 6, 8]**
`print(aList2)`	**[1, 2, 4, 5]**
`print(aList3)`	**[10, 12, 14]**

```
main()
```

Problem 6) [10 marks]

Write a complete function called findDivisors that accepts any positive int value, one or higher, and then returns a list of integers that divide the supplied number evenly, including one and the number itself. The resulting list should be in ascending order. If the supplied number is less than 1, return an empty list. Remember that the append() method appends a number to the end of a list.

For example, if findDivisors is invoked with the number 110, it would return the list:
[1, 2, 5, 10, 11, 22, 55, 110].

```
def findDivisors(aNum) :

    if aNum < 1 :
        return [ ]

    aList = [ ]

    for divisor in range(1, aNum) :
        if aNum % divisor == 0 :
            aList.append(divisor)

    aList.append(aNum)

    return aList
```

Problem 7) [18 marks]
The selection sort algorithm works using a nested loop structure. The outer loop moves from the beginning of the list to be sorted towards the end. If the list is to be sorted into increasing order, the inner loop searches the unsorted portion of the list to find the smallest remaining element in the list. Once found, that element is swapped with the element at one position past the sorted portion of the list. This process continues until the outer loop is one position short of the end of the list, at which point the entire list is sorted.

For this problem you must apply the selection sort algorithm to work on a list of lists (or a *database* of *records*). Your function will accept the list to be sorted and two field numbers as parameters, field1 and field2. The field numbers are positions inside the smaller lists (or *records*), which are inside the big list (or *database*) to be sorted. You sort the list based on the value in field1 as usual, but if you get two records that have the same values in field1 then you compare the two records based on field2. (Don't worry about what happens if the values in field2 are also the same!) A telephone book uses this same procedure, sorting phone records based on last name and then by first name when the last names are equal. As a further example, consider the following set of records which could be first name, last name and age:

[['Joe', 'Brown', 42], ['Mary', 'Brown', 23], ['Harry', 'Brown', 32], ['Joe', 'Green', 42], ['Bob', 'Brown', 29], ['Spam', 'Alot', 42], ['Monty', 'Python', 39]]

If field1 and field2 are 1 and 0 respectively (last name followed by first name), then the sorted set would be:

[['Spam', 'Alot', 42], ['Bob', 'Brown', 29], ['Harry', 'Brown', 32], ['Joe', 'Brown', 42], ['Mary', 'Brown', 23], ['Joe', 'Green', 42], ['Monty', 'Python', 39]]

Or, if field1 is 2 and field2 is 1 (age followed by last name) then you would get:

[['Mary', 'Brown', 23], ['Bob', 'Brown', 29], ['Harry', 'Brown', 32], ['Monty', 'Python', 39], ['Spam', 'Alot', 42], ['Joe', 'Brown', 42], ['Joe', 'Green', 42]]

Write your swap function here:

```
def swap(dbList, pos1, pos2) :

    temp = dbList[pos1]
    dbList[pos1] = dbList[pos2]
    dbList[pos2] = temp
```

Problem 7, Cont.)
Write your selection sort function here. Your function must use the swap function you wrote on the previous page. Note that if either field value is illegal your function should raise a ValueError exception. Your sort function should work for a database of any size (but you can assume it will not be empty) and records of any length greater than 1 (which you can also assume). You can also assume that each record in the database will have the same format and number of elements.

```
def selectionSort(dbList, field1, field2) :

    if field1 > len(dbList[0]) - 1 or field2 > len(dbList[0]) \
      or field1 < 0 or field2 < 0 :
       raise ValueError("Illegal field number")

    i = 0
    size = len(dbList)

    while i < size - 1 :
        smallestPos = i
        j = i + 1
        while j < size :
            if dbList[j][field1] < dbList[smallestPos][field1] :
                smallestPos = j
            elif dbList[j][field1] == dbList[smallestPos][field1] :
                if dbList[j][field2] < dbList[smallestPos][field2] :
                    smallestPos = j
            j = j + 1
        if smallestPos != i :
            swap(dbList, i, smallestPos)
        i = i + 1
```

Problem 8) [12 marks]

The Saffir-Simpson Hurricane scale gives a hurricane its category rating largely depending on its average wind speed:

Category:	1	2	3	4	5
Wind Speed (km/hour)	119-153	154-177	178-209	210-249	> 249

Here is the beginning of a GUI program that is designed to tell the user the category of a hurricane once he provides the wind speed and clicks on a button. There is enough code to allow you to figure out the layout of the widgets on the window. Draw the approximate appearance of the window in the blank window below. *Don't worry about imitating fonts or using a ruler!*

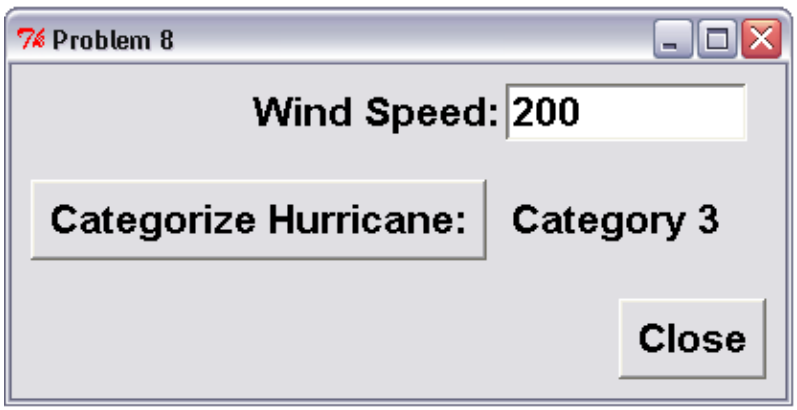
```
import Tkinter

top = Tkinter.Tk()
windSpeedEntry = Tkinter.Entry(top, width=10, font="Arial 16")
categoryLabel = Tkinter.Label(top, text="", font="Arial 16")

def main() :

    top.title("Problem 8")
    categorizeButton = Tkinter.Button(top, text="Categorize Hurricane:", \
                            font="Arial 16 bold", command=categorize)
    quitButton = Tkinter.Button(top, text="Close", font="Arial 16", \
                              command=top.quit)
    windSpeedLabel = Tkinter.Label(top, text="Wind Speed:", font="Arial 16")
    windSpeedLabel.grid(row=0, column=0, sticky="e")
    windSpeedEntry.grid(row=0, column=1, sticky="w")
    categorizeButton.grid(row=1, column=0)
    categoryLabel.grid(row=1, column=1, sticky="w")
    quitButton.grid(row=2, column=1, sticky="e", padx=10)
    top.columnconfigure(0, pad=20)
    top.columnconfigure(1, pad=20)
    top.rowconfigure(0, pad=20)
    top.rowconfigure(1, pad=20)
    top.rowconfigure(2, pad=20)

    Tkinter.mainloop()
```



Use these shapes to show your widget positions:

A Label:

An Entry Box

A Button

Problem 8, Cont.)

To complete this program, you need to write the missing function. It will take the entry box's wind speed and display the category of the hurricane in the appropriate label. Assume that the user will always enter a valid integer. Remember that you can use the get() method to get the contents of an entry box (returned as a string), and you can set the text of any label by invoking the configure() method as in *label*.configure(text = labelString), where labelString is the string you wish displayed. Modify your drawing on the previous page to show what the window would look like if the user typed "200" into the entry box and clicked the categorize button.

```
def categorize() :

    windSpeed = int(windSpeedEntry.get())

    if windSpeed < 119 :
        result = "Not a Hurricane!"
    elif windSpeed >= 119 and windSpeed <= 153 :
        result = "Category 1"
    elif windSpeed >= 154 and windSpeed <= 177 :
        result = "Category 2"
    elif windSpeed >= 178 and windSpeed <= 209 :
        result = "Category 3"
    elif windSpeed >= 210 and windSpeed <= 249 :
        result = "Category 4"
    else :
        result = "Category 5"

    categoryLabel.configure(text=result)
```

(*blank page*)