Please fill in your Student Number and Name.

Student Number : ________________________

Name:

____

____

Student Number:

____

University of Cape Town ~ Department of Computer Science

Computer Science 1015F ~ 2012

June Examination

*** SOLUTIONS ***

Question	Max	Internal	External
1	5		
2	7		
3	18		
4	20		
5	10		
6	10		
7	10		
8	20		
TOTAL	**100**		

Marks : 100

Time : 180 minutes

Instructions:

a) Answer all questions.

b) Write your answers in PEN in the spaces provided.

c) Show all calculations where applicable.

Question 1 Multiple Choice [5]

For each of the multiple choice questions below, write down the letter corresponding to the correct answer. Note that all code examples refer to Python version 3.

(a) The **Fetch-Execute** cycle is:

i. important for compilers, but not interpreters.

ii. is used for low-level languages, but not high-level languages.

iii. Is the method of designing an algorithm to solve a particular problem

iv. is a process followed by the CPU when running a program.

D

(b) In Python 3, the expression `print("to","to","ro",sep='\n')` will evaluate as:

i. to\nto\nro

ii. tontonro

iii. to
to
ro

iv. totoro

C

(c) `type(3.0)` will evaluate as:

i. NameError

ii. <class 'str'>

iii. <class 'int'>

iv. <class 'float'>

C

(d) The expression `"nausicaa"[1:7:3]` will evaluate as:

i. 'ai'

ii. 'asc'

iii. 'nsa'

iv. 'aia'

A

(e) `4>3 or 'happy' and 'sad'` will evaluate as:

i. True

ii. 'happy'

iii. 'sad'

iv. TypeError

A

Question 2 [7]

(a) Explain clearly and concisely the difference between **machine languages** and high-level languages and the **purpose** of an **interpreter.** [4]

The language understood by a computer is called a machine language. It is in binary.[1] High level languages use recognisable words and are more easily understood by humans. They are portable. [1] An interpreter is a complex program that translates [1] a program written in a high-level language[1] into immediately executable machine code [1].

(b) Python has **two** basic numeric data types. What are they called and why is it it necessary to have two separate types for numbers? [3]

int and float[1] Integer and floating point arithmetic is handled differently in the hardware. [1] Integer arithmetic is faster and therefore it makes sense to use the integer arithmetic when possible. [1]

Question 3 [18]

Examine the module listed below:

```
def bob(s):
    m=0
    c=''
    for x in s:
        tally=s.count(x)
        if tally>m:
            m=tally
            c=x
    print(c,m)

bob("totoro")
bob("cat")
bob("")
```

(a) From the module listed above, give an example of:

i. a variable [1]

Any of the variables used in the program: x, tally etc.

ii. a literal [1]

Any of the strings (“totoro”, etc) used in the program

iii. a string function [1]

count

iv. a relational operator [1]

>

(b) Explain, in general terms, what the function `bob` does. [2]

It searches through a string to find the highest frequency character and outputs the number of times it occurs

(c) Write down the **exact** output of this module when it is run in the Python interpreter (i.e. when you press “Run” in the Wing IDE). [6]

t...o...3

c...1

0

(d) The function `bob` uses a definite loop. Rewrite the function so that it behaves exactly the same, but uses an indefinite loop. [3]

```
def bob(s):
    m=0
    c=''
  while s: #one for this while line
    x=s[0]  #one for setting the initial loop counter
    tally=s.count(x)
    s=s[1:]  #1 for reducing the string length
    if tally<=m: continue
    m=tally
    c=x
    print(c,end="...")
  print(m)
```

(e) When called with a string (like "lolololololo"), the algorithm in function `bob` is not very efficient. Explain why this is and suggest how you could improve the algorithm. [3]

With a string like "lolololololo", the function bob will call the count function once for every time a letter is repeated. One improvement could be to use the replace function to remove a character from the string once it is counted.

Question 4 [20]

Student travel poll

Varsity newspaper has hired you to perform a poll of the students on campus, to find out how students travel to campus. Someone has claimed that Science students use the Jammie Shuttle more than students in Arts, Engineering, Commerce or Law, and Varsity would like you to check whether this is true. Varsity would also like to report on what percentage of the whole student body travels by Jammie Shuttle, car, train, on foot or by other means. You have decided to write a simple Python program to assist you in this task of collecting data from students.

(a) Do you think that a sentinel loop would be a good idea for this program? Justify your answer, explaining the concept of a sentinel loop. [3]

Yes. [1] A sentinel loop is an indefinte loop that continues until a certain "sentinel" value (non-data) is entered [1]. This loop would be useful as you don't know how many students you will interview.[1]

(b) Write the Python code for your application. You do not need to do extensive error checking in this code – just write the basic algorithm and assume that the input data will always be correct. [12]

```
#There are a number of correct answers, of course
#could use lists/dictionaries also
S_JS,c,f,J,t,o=0,0,0,0,0,0 #1 for intializing counters
total=0
faculty= input("Science faculty? (y/n)")


while faculty:
   travel=input("Travel by car(c),foot(f),Jammie Shuttle(j),train(t) or other (o)?") #1
   if travel=='c':
      c+=1
   elif travel =='f':
      f+=1
   elif travel =='j':
      J+=1
      if faculty=='y':
         S_JS+=1
   elif travel =='t':
      t+=1
   elif travel == 'o':
      o+=1
   total+=1
   faculty= input("Science faculty? (y/n)")
```

```
print('J=', J/total*100, 'c=',c/total*100,'t=',t/total*100,'f=',f/total*100,'o=',o/total*100)
print('Science J=', S_JS/total*100)  #4 suitable output
```

(c) *Varsity* now wants to write a fun article on the most and least common birth dates on campus. Can you reuse the same algorithm/approach that you employed in your solution above, or would you need to change your approach? Justify your answer. [3]

The answer depends, obviously, on which algorithm they used The one above would not be suitable, as the elif ladder would be very tedious to write for birthdates. An array/dictionary would then be a better plan.

(d) *Varsity* now also wants to commission an online dating service and would like you to write the code that will match potential partners. You go to your lecturer for advice and she says that, from a Computer Science point of view, it would be a very bad idea to do this. Explain her reasons, clearly and briefly. [2]

For this code you will need a clear algorithm. However, there is not really an algorithm for dating that has been proven to work.

Question 5 [10]

Tic-Tac-Toe (aka Noughts and Crosses) is a game where 2 players, assigned the symbols X and O, take turns to place their symbols on a 3x3 grid, with the winner defined as the first player to create a horizontal, vertical or diagonal line of 3 symbols.

The following is a typical game board after 4 turns:

X	O	X
	O	

(a) Write a set of Python statements to create a 3x3 2-dimensional array of “ ” (space) characters in the variable *grid*. [3]

```
grid = [] ... [1]
for i in range(3): ... [1]
  grid.append ([' '] * 3) ... [1]
```

(b) Write a function `checkBoard (grid)` that returns the symbol of the winner or “ ” (space) if the board is still incomplete. [7]

```
def checkBoard (grid):
  for r in range (0, 3): [2]
    if grid[r][0]==grid[r][1]==grid[r][2]:
      return grid[r][0]
    if grid[0][r]==grid[1][r]==grid[2][r]: [2]
      return grid[0][r]
  if grid[0][0]==grid[1][1]==grid[2][2] or grid[1][1]==grid[0][2]==grid[2][0]:  [2]
      return grid[1][1]
  return ' ' [1]
```

Question 6 [10]

As a holiday job, you have been assigned the task of stock control in your mum's spaza shop. Being a typical computer science student, you first entered the stock number of each item into a program. Each time a product is ordered, you have to search for the item. You have decided that it is more efficient to use a binary search but you first need to sort the list in ascending order.

The list of items is as follows:

12	3	11	2	7	5	13	8

(a) Show how the quicksort algorithm can be applied to this list to sort the items in ascending order. Show each step of the algorithm. Use the last item as the pivot. Clearly indicate the list(s) and the pivot at each step (using [] around a list and () around a pivot). [4]

[3 2 7 5 (8) 12 11 13]

[3 2 (5) 7] 8 [12 11 (13)]

[(2) 3] 5 [7] 8 [(11) 12] 13

2 [3] 5 7 8 11 [12] 13

2 3 5 7 8 11 12 13

(b) Your mum thinks that you are wasting your time and should instead just use a linear search. Explain, using known information about the efficiency of each algorithm, why your solution is faster, even for only 8 items. [2]

Each time linear search is O(n); sorting first is O(n log n) and then each binary search is O(log n)[1]. for many search operations, the searching takes much more time than the sorting, so the second approach is faster [1].

(c) Write a **recursive** function to check if an item is in an unsorted list. Your function must return 0 if the item is not found and 1 if it is found. [4]

```
def linsearch (item, itemlist):
```

```
def linsearch (item, itemlist):
    if len(itemlist)==0: [1]
        return 0
    elif itemlist[0]==item: [1]
        return 1
    else:
        return linsearch (item, itemlist[1:]) [2]
```

Question 7 [10]

(a) Convert the decimal number 42 to binary, showing your working. [2]

42/2 = 21 rem 0

21/2 = 10 rem 1

10/2 = 5 rem 0

5/2 = 2 rem 1

2/2 = 1 rem 0

1/2 = 0 rem 1 answer: 101010

(b) Convert the octal number 312 to hexadecimal, showing your working. [2]

011|001|010 = 0|1100|1010

answer: CA

(c) Calculate 12 - 2 using 8-bit 1's complement binary representation, showing all your working. (i.e. convert the numbers to 8-bit 1's complement before adding). Leave your final answer in 1's complement form. [3]

00001100 – 00000010

= 00001100 + 1s(00000010)

= 00001100 + 11111101

= 00001011 carry 1

= 00001011 + 1

= 00001010

(d) Find the value represented by the floating point number written below using IEEE754 representation (bias of 127 added to the exponent; first bit is the sign bit, next 8 bits are the biased exponent, rightmost 23 bits are the significand). Show all your working. [3]

```
1 10000000 01100000000000000000000
```

sign implies negative value

biased exponent is 128 so actual exponent is 128 – 127 = 1

mantissa is 1.011 in binary which is 1(.125+.25) =1.375 in decimal

*so number is - 1.375 * 2^1 = - 2.75*

Question 8 [20]

The Event Manager

As a young student at UCT, you have discovered numerous extra-curricular activities that you would like to participate in. However, it is difficult to keep track of the many events taking place around campus. You have therefore decided to store a list of events in a text file, with multiple programs to update the file and search through the file.

The "data.txt" text file will have the following format:

```
seminar,What's at the bottom of the sea,Spongebob Squarepants,2012-05-08,16:00,17:00

movie,My favourite colour: Blue,Papa Smurf,2012-05-08,18:00,20:00

debate,Who is the fairest of them all,The seven dwarves,2012-05-12,18:00,19:00
```

The *add.py* program, to add entries to the file, is listed below:

```
details = input ("Enter new event details:")
key = ",".join ((details.split (","))[3:5])      # line 2
f = open ("data.txt", "r")
data = f.readlines()
f.close ()

position = 0
filekey = ",".join ((data[position].split (","))[3:5])
while position<len(data) and filekey<key:
   position+=1
   if position<len(data):
      filekey = ",".join ((data[position].split (","))[3:5])

data.insert (position, details+"\n")             # line 14
f = open ("data.txt", "w")
for line in data:
   print (line, file=f, end="")
f.close ()
```

(a) Explain in simple English what the purpose of line 2 of *add.py* is. [2]

The date and time are extracted from the new details [1] and concatenated into a single string [1]

(b) In line 14 of *add.py*, an extra newline is added to the new entry. Explain why. [2]

all existing entries already have newlines and newlines are not printed when the file is updated so the newline has to be added beforehand[2]

(c) If we are counting comparisons, what is the best case time complexity (order) of the *add.py* program? What is its worse case time complexity? [1]

O(1) [1]; O(n) [1]

(d) When defining equivalance classes based on the size of the existing file, at what boundary value does *add.py* not work? Explain why. [2]

empty file [1]

the program tries to access data at position 0 without checking if it exists [1]

(e) The date is in Year-Month-Day format. Will *add.py* still work if it is in Day-Month-Year format? Explain your answer. [2]

No. The comparison of dates will fail because days will be considered first so a difference in days will be more significant than a difference in months or years. [2]

(f) Write a program to ask the user for a date, read in the entire file and print out all matching entries. Ignore the time values. [8]

```
details = input ("Enter date:") [1]
f = open ("data.txt", "r") [3]
data = f.readlines()
f.close ()
for entry in data: [4]
    key = (entry.split (","))[3]
    if key==details:
        print (entry,end="")
```

(g) Describe a more efficient algorithm than reading in the entire file in the previous question. [2]

Read in one line at a time [1] and stop after the date becomes larger than the date being searched for (or we reach end of the file). [1]