ICT133

Examination – July Semester 2021

Structured Programming

Thursday, 18 November 2021 **1:00 pm – 3:00 pm**

Time allowed: 2 hours

Marking Guide

Strictly Confidential

Under no circumstances should students be allowed to see the contents of this book. Should this book, or any part of it, fall into the hands of a student, SUSS must be informed immediately.

General advice to markers

1. Do not deviate from the marking guide it has been vetted and approved by the Head of Programme.

2. All markings should be completed in ink.

3. Initial against all corrections in markings.

4. Do not award fractional marks (i.e. half mark).

5. Never mark on a keyword basis, namely awarding marks by spotting words similar to those in the answer guide. If the answer is meaningless, no mark should be awarded even though similar words appear in the suggested answer in the marking guide.

6. Where students are given a choice of questions to attempt and if a student has answered more than the required number of questions:
   a) Mark only those question numbers that the student has indicated as having attempted on the cover page of the answer book; and
   b) If there are more questions indicated on the cover page than there are attempted in the answer book, mark the answers sequentially as they appear in the answer book until the required number of questions; the remaining answers thereafter should not be marked.

Learning Outcomes

Learning Outcome Assessed		EQP			
		Q1	Q2	Q3	Q4
LO1	Describe the building blocks of computers and programs	✓			
LO2	Express a sequence of statements based on computational logic	✓			
LO3	Apply data structures to store and process information			✓	✓
LO4	Employ structured programming principles to develop a program		✓		
LO5	Develop an application to address practical requirements			✓	✓
LO6	Solve computational problems using structured programming		✓		✓

Question 1 (25 marks)

Describe and apply control structures based on computational logic.

(a) Given the following code segment:

```
if x > y:
   if z < y:
      print(y)
   elif x < z:
      print(x)
   else:
      print(z)
else:
   if y < z:
       print(y)
   elif x < z:
       print(z)
   else:
       print(x)
```

(i) What is the output if the values of x, y and z are 1, 3, 2?

(2 marks)

(ii) What does the code segment do for different values of x, y, and z?

(3 marks)

(b) Write and **express** a program that converts a 24 hour clock time to a 12 hour clock time. For example, 2359 is 11.59PM and 0910 is 9.10AM. Input is a string representing the 24 hour clock. No validation is required.
Two sample output are as follows:

Sample 1
```
Enter 24 hour time: 1305
Time in 12 hour format is 1.05PM
```

Sample 2
```
Enter 24 hour time: 0010
Time in 12 hour format is 12.10AM
```

(10 marks)

(c) Write a function `validateInput(time)` that has a string parameter. The function returns `True` if the time parameter is a valid 24 hour time, and `False` otherwise. A valid 24 hour time satisfies the following conditions:
- length is 4
- contains all digits
- the first 2 digits that represent the hour must be between 0 and 23 inclusive
- the last 2 digits that represent the minutes must be between 0 and 59 inclusive

(10 marks)

MARKING GUIDE:

(a) (i)

```
2
```

- 2 marks or nothing.

(Total 2 marks)

(ii)

```
It prints the middle value for all different values of x, y, z.
```

- 3 marks for correct explanation, no partial marks.

(Total 3 marks)

(b)

```
clock = input('Enter clock in 24 hr: ')
hr = int(clock[0:2])
min = int(clock[2:])
if hr == 0:
    hr = 12
    session = 'AM'
elif hr > 12:
    hr -= 12
    session = 'PM'
elif hr == 12:
    session = 'PM'
else:
    session = 'AM'
print( f'{hr}.{min:02d}{session}')
```

- 1 mark for input as string
- 2 marks for extracting hour, min and convert to int
- 5 marks for getting the hour and session correct.
  - each mistake minus 2 marks. E.g. hour incorrect, or session incorrect
- 2 marks for correct printing format
  - award only 1 mark if minutes not in 2 digits.

Award up to maximum of 4 marks, if there are 2 or more errors in the checking.

(Total 10 marks)

(c)

```
def validateInput(time):
    if len(time) != 4 or not time.isdigit():
        return False
    hr = int(time[0:2])
    mins = int(time[2:])
    if not 0<=hr<24 or not 0<=mins<=59:
        return False
    return True
```

- 1 mark function header
- 2 marks each for checking:
  - length is 4
  - only digits
  - first 2 digits between 0 and less than 24
  - second 2 digits between 0 and less than 60
- 1 mark for return
- If there are 2 or more errors in the checking, do not award more than 4 marks overall.

(Total 10 marks)

Question 2 (25 marks)

Interpret and **solve** computational problems using structured programming.

(a) What is the output of the following code segment?

```
x = 1
y = 2
for n in range(5):
   z = x + y
   x = y
   y = z
   print(z, end=' ')
```

(5 marks)

(b) The following function `replace` has a string parameter and replaces all occurrences of `char1` with `char2`. Both `char1` and `char2` are single characters.

```
1.  def replace(string, char1, char2):
2.      for n in range(len(string)):
3.          temp = ''
4.          if string[n] == char1:
5.             temp = char2 + temp
6.          return temp
7.
8.  print(replace('ajax', 'a', 'A'))
```

The function is tested with the statement in line 8. Examine the code and answer the following questions.

(i) How many times will the for loop in line 2 be executed?

(2 marks)

(ii) What output will be displayed?

(2 marks)

(iii) Rewrite the function to fix all logic errors so that the output will display as `AjAx`.

(6 marks)

(c) Write and **employ** a function `index(string, substring)` that has 2 string parameters.

You are not allowed to use the inbuilt string methods `index()`, `find()` in Python, and you may not convert string to a list.

The `index()` function returns the starting index position of a matching substring in the string. If there is no match, return -1. E.g. if the string is ‘vaccination’, the following cases will return values as:

- substring ‘a’ returns index 1, the first ‘a’ found in the string
- substring ‘cc’ returns index 2, the start index of the string that matches the substring
- substring ‘ace’ returns -1, as there is no match
- substring ‘vaccinations’ returns -1

Assume that the string and substring are all lowercase letters.

(10 marks)

<u>MARKING GUIDE:</u>

(a)

```
3 5 8 13 21
```

- 1 mark each for correct numbers in the correct sequence
- Minus 1 mark if output is one number per line.

(Total 5 marks)

(b)

(i)

```
Once
  -   2 marks or nothing
```

(ii)

```
A
  -   2 marks or nothing
```

(iii)

```
def replace(string, char1, char2):
    temp=''
    for n in range(len(string)):
        if string[n] == char1:
            temp += char2
        else:
            temp += string[n]
    return temp
```

- 1 mark for temp outside loop
- 4 marks for if … else loop, and appending the letters correctly to temp.
  - Minus 2 marks per mistake.
- 1 mark for return

(Total 6 marks)

(c)

```
def index(string, substr):

    for n in range(length(string)-length(substr)+1):
        match = True
        for m in range(length(substr)):
            if substr[m] != string[n+m]:
                match = False
                break
        if match:
            return n
    return -1
```

- 10 mark, 2 marks each for each checking:
  - checking of first letter match
  - checking of more than 1 letter match
  - checking of length of substring longer than

Award up to maximum of 5 marks if the logic does not make sense, e.g. will cause runtime error.

(Total 10 marks)

Question 3 (25 marks)

Apply data structures to solve computational problems.

(a) **Develop** and write a function `count(numlist)` that has a list of integers as parameter. The function returns a count of the occurrences of each number in the list as a dictionary. E.g. given a list of numbers:
`numlist = [2, 1, 3, 6, 2, 1, 4, 1, 2, 6, 5, 1, 1, 2]`

the function returns `{2: 4, 1: 5, 3: 1, 6: 2, 4: 1, 5: 1}`, since the value 2 occurs 4 times, the value 1 occurs 5 times etc. If `numlist` is empty, an empty dictionary is returned.

(13 marks)

(b) Write a function `mode(numlist)` that has a list of integers as parameter. Mode is the value that occurs the most times in a list. The function returns the number in the list which occurs most frequently. E.g. given the list in part Q3(a), the function returns a list with value [1], since 1 occurs the most time. It is possible that there may be more than one mode. In this case, return the mode as a list, as in [2, 5], assuming 2 and 5 occurred the most time in the list. Make use of the function in part Q3(a).

(12 marks)

MARKING GUIDE:

(a)	
	```
	def count(numlist): #count
	    numdict = {}
	    for num in numlist:
	        if num not in numdict:
	            cnt = 0
	            for m in numlist: #count function also accepted
	                if num == m:
	                    cnt += 1
	            numdict[num] = cnt
	    return numdict
	```
	 - 1 mark for function header
	- 1 mark for create dictionary
	- 5 marks for looping the list and checking the key against dict
	- 5 marks for counting the occurrences
	- 1 mark for return
	Award up to a maximum of 7 marks if the logic for counting is incorrect.
	(Total 13 marks)

(b)	
	```
	def mode(numlist):
	    temp = []
	    numCount = count(numlist)
	    maxValue = max(numCount.values())
	    for n, v in numCount.items():
	        if v==maxValue:
	            temp.append(n)
	    return temp
	```
	 - 1 mark to declare empty list

	- 2 marks for counting (make use of count function) - 3 marks for finding the maximum value for the values in the dictionary. - 3 marks for loop over count dictionary - 2 marks to check for maximum and appending the key with max values in the dictionary. - 1 mark for return Award up to a maximum of 6 marks if the logic to find the mode is incorrect. (Total 12 marks)

Question 4 (25 marks)

Apply data structures to store and process information. **Employ** structured programming principles and **solve** computational problems.

A university admits students based on the total points for 4 subject grades. The applicants' grades are stored in a file. A program reads the applicants grades from a file and calculates the points based on the grades and prints a summary report on the admission eligibility. The points awarded based on the grade is given by the following table:

Grade	Points
A	20
B	16
C	12
D	10
E	6

Table Q4. Grade point table

Total points of 50 and above are considered for admission. The applicant file consists of student id, and 4 subject grades, one student per line. Refer to appendix A for the file contents.

The program consists of functions as follows:

(a) Write a function `calculatePoints(grades)` that has a grades parameter. The grades parameter is a list of grades, as in `['A', 'B', 'C', 'C'].` The function returns the total points based on the grade point table. No selection statements are allowed in this function.

(6 marks)

(b) Write a function `readFromFile(filename)` that reads applicants grades from a filename parameter. The function returns a dictionary in the following format:

`applicants={'S1':['A','A','A','C'], 'S2':['B','B','C','C']}`
(only 2 shown). The key is student id, and the value is a list of grades. The grades should be stripped of trailing blanks and \n characters.

(8 marks)

(c) Write a function `printEligibilityReport(applicants)` that has a dictionary of applicants as parameter. The function prints a report of the points attained for each student and their eligibility for admission to the university. Students with total points 50 or more are admitted. A sample report is as follows:

```
S1 A,A,A,C 72 pts Admitted
S2 B,B,C,C 56 pts Admitted
S3 D,D,A,E 46 pts Rejected
S4 B,C,D,A 58 pts Admitted

3 admitted, 1 rejected.
```

(7 marks)

(d) Write a main function that reads in the file and prints the eligibility report.

(4 marks)

MARKING GUIDE:

(a)

```
def calculatePoints(grade):
    points= {'A':20, 'B':16, 'C':12, 'D': 10, 'E': 6}
    return sum([points[v] for v in grade])

 -  2 marks for points declaration
 -  3 marks for the sum logic and return
 -  1 mark for return
 Minus 3 marks if selection statements used
```

(Total 6 marks)

(b)

```
def readFromFile(filename):
    applicants = {}
    with open(filename) as fr:
        for lines in fr:
            students = lines.split(',')
            for n in range(1, len(students)):
                students[n] = students[n].strip()
            applicants[students[0]] = students[1:]
    return applicants

 -  1 mark for declaration of dictionary
 -  1 mark for opening file for reading
 -  2 marks for loop to read each line from file
 -  3 marks for logic to strip and  populate dictionary
 -  1 mark for return
 Award upto a maximum of 4 marks if the overall logic is incorrect.
```

(Total 8 marks)

(c)

```
def printEligibilityReport(applicants):
    for stud, grades in applicants.items():
        pts = points(grades)
        if pts >= 50:
            print(f'{stud} {",".join(grades)} {pts} Admitted')
        else:
            print(f'{stud} {",".join(grades)} {pts} Rejected')

 -  3 marks for loop over dictionary
 -  2 marks for getting points using points function
 -  2 marks for if else logic to print
```

(Total 7 marks)

(d)	
	```
	def main():
	    applications=readFromFile('applicants.txt')
	    printEligibilityReport(applications)
	```
	 - 2 marks for each function call
	(Total 4 marks)

----- END OF MARKING GUIDE -----