THE UNIVERSITY OF WESTERN AUSTRALIA

DESK No. | | | |

FAMILY NAME: _____

GIVEN NAMES: _____

SIGNATURE: _____

STUDENT NUMBER: | | | | | | | |

SEMESTER 1, 2021 EXAMINATIONS

CITS1401

Computational Thinking with Python
Examination Solution

Physics, Mathematics & Computing
EMS

This paper contains: 10 Pages **(including title page)**

Time Allowed: **2:00** hours

INSTRUCTIONS:

- This examination paper comprises of 1 section containing 8 questions.
- Attempt all questions and total marks are 100.
- Answer to all the questions are to be written in the spaces provided in this exam booklet.
- Use the blank pages in the start and at the end for rough work.
- No calculators or other aids are allowed.
- You must put your name and number on this page and any other booklet you use during the exam. All these must be handed in at the end of the exam.

Office use only.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8

THIS IS A CLOSED BOOK EXAMINATION

SUPPLIED STATIONERY	**ALLOWABLE ITEMS**

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Candidates must comply with the Examination Rules of the University and with the directions of supervisors.
No electronic devices are permitted during the examination.
All question papers and answer booklets are the property of the University and remain so at all times.

There can be multiple solutions but only one is provided for understanding purpose.

This page has been left intentionally blank

There can be multiple solutions but only one is provided for understanding purpose.

Library modules are not allowed to be imported for any of the question in this exam.

1. Define a Python function `stringexpand(name)` which, given a string `name`, returns the expanded string such that white space is added after each character of the input string. For example, `stringexpand("Adam")` returns "A d a m". **[5 Marks]**

```
def stringexpand(name):
    ne = ""
    for ch in name:
        ne += ch + " "
    return ne[:-1]
```

2. Define a Python function `palindrome(word)`, which checks whether the input string is palindrome or not (irrespective of case sensitivity) and then returns boolean result accordingly. Palindrome string is the string which is same both backwards and forwards. For example, `palindrome("Racecar")` will return `True`. Similarly, `palindrome("dude")` will return `False`. **[5 Marks]**

```
def palindrome(word):
    word = word.lower()
    for i in range(len(word)):
        if word[i-1] != word[-i]:
            return False
    return True
```


There can be multiple solutions but only one is provided for understanding purpose.

3. Define a Python function `sortdict(dict)` which, given a dictionary `dict` of student IDs and students WAM as keys and values respectively, returns the list of tuples sorted by student WAM in descending order. For example,
`sortdict({212222:35,2222222:85,2232222:55})` returns
`[(2222222, 85), (2232222, 55), (212222, 35)]`
Hint: Think about using sort() function with lists. **[10 Marks]**

```
def sortdict(dict):
    stlist = []
    for key in dict:
        stlist.append( (key,dict[key]) )
    stlist.sort(key = lambda x:x[1], reverse=True)
    return stlist
```

4. [Use recursion to solve this problem]
Define a Python function `sumdigits(n)` which, given a positive integer, returns the sum of the digits of input `n`.
For example, `sumdigits(1234)` returns `10` (obtained by summing all the digits of input: 1+2+3+4=10).
[10 Marks]
Note: Recursion must be used to solve this question. Looping is not allowed.

```
def sumdigits(n):
        if n == 0:
                return 0
        sumdig = n % 10 + sumdigits(n//10)
        return sumdig
```


There can be multiple solutions but only one is provided for understanding purpose.

5. Below mentioned is the Maclaurin series for the exponential function e^x. Summing higher number of terms provides better approximation.

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots = e^x$$

where "!" represents the factorial such as 1!=1,2!=2 x 1=2,3!=3 x 2 x 1=6 and so on.

Define a Python function `seriesexp(x,tol)` which, given numerical values `x` and `tol`, returns the sum of the above series for input `x` till the absolute value of the term becomes smaller than `tol`. For example, `seriesexp(1,0.1)` returns `2.66666` which is obtained by adding first four terms only because the absolute value of fifth term $^{1^4}/_{4!}$ becomes smaller than the input to the function 0.1 i.e. (1/24)<0.1

[15 Marks]

```
def seriesexp(x,tol):
    term = 1
    series = 0
    i = 1
    while tol <= term :
        series += term
        term = x**i/factorial(i)
        i += 1
    return series
def factorial(n):
    if n < 2:
        return 1
    fact = 1
    for i in range(n,1,-1):
        fact *= i
    return fact
```


There can be multiple solutions but only one is provided for understanding purpose.

6. Write a Python function `multiply_lists(lst)` which can return the product of the numerical data in the input list `lst`. However, it is possible that the input list, `lst`, possibly contain other lists (which can be empty or further contain more lists). You can assume that the lists only contain numerical data and lists. For example, `multiply_lists([1,2,[],3.5,4])` returns 28.0;
Similarly `multiply_lists([1,[2],[3.5,[4]]])` also returns 28.0.

[20 Marks]

```
def multiply_lists(lst):
    if len(lst) == 0:
        return 1
    total = 1
    for item in lst:
        if type(item) == type(10) or type(item) == type(10.01):
            total *= item
        else:
            total *= multiply_lists(item)
    return total
```


There can be multiple solutions but only one is provided for understanding purpose.

7. Claremont cricket club have many players and need to make combination of three players for the practice sessions. There are three type of players: Batsmen, bowler and wicket-keeper. All of them are in different pools and the club need to select one player from each pool and make a combination. All players in each pool need to have practice session with all players of other pool.
   Write a python function `team_maker(pool1,pool2,pool3)` for the above situation which, given three lists of pools of players `(pool1,pool2,pool3)` returns a list containing all different combinations for the practice sessions. Each item of the output list has the names of players combined as a string with a white space " "
   For example,
   `team_maker(['Smith','Finch'],['Cummins','Lyon'],['Adam','Payne'])`
   returns `['Smith Cummins Adam', 'Smith Cummins Payne', 'Smith Lyon Adam', 'Smith Lyon Payne', 'Finch Cummins Adam', 'Finch Cummins Payne', 'Finch Lyon Adam', 'Finch Lyon Payne']`

   [15 Marks]

```
def team_maker(pool1,pool2,pool3):
    teams = []
    for i in range(len(pool1)):
        for j in range(len(pool2)):
            for k in range(len(pool3)):
                teams.append(pool1[i] + " " + pool2[j] + " " + pool3[k])
    return teams
```


There can be multiple solutions but only one is provided for understanding purpose.

8. Write a Python function `user_account(filename)` which, given a file name reads the file, then extracts the users' first name, last name and date of birth, creates the login and passwords for the user and stores them in `logins.txt`. The file filename will be a text file in which each value is separated by a comma "," character - and each 'row' of values is separated by a new line ("\n") at the end. Each extracted login and password is stored in `logins.txt` in such a way that the values are separated by a comma "," and each user data is stored in a separate line. The order of the user data is according to the title of the column, however the order of the titles is not known. Therefore, you must need to search the indexes of titles "First name", "Last name" and "Date of birth" and use these indexes in retrieving user's data.

   The methodology for creating the login is by combining first and last name whereas dot "." is between them. The passwords are created by last name followed by a dollar sign "$" which is followed by digits of date of birth.

   For example, a file `"userdata.txt"` contains following five lines of text

```
s_no,First name,Middle name,Last name,Date of birth,Address,Phone
1, Ali,Kem,Khan,12/5/2000,Crawley WA,+43444444
2, John,Roger,Smith,1/12/1988,Perth City 6000,+61821021010
3, Sarah,Kimberly,Jones,22/8/1999,Subiaco WA,+61431312312
4, Huan,Jian,Li,5/8/2002,Claremont 6010 WA,+618323454
```

   Running the function `user_account("userdata.txt")` creates the file `login.txt` having following contents.

```
Ali.Khan,Khan$1252000
John.Smith,Smith$1121988
Sarah.Jones,Jones$2281999
Huan.Li,Li$582002
```

[20 Marks]

Note: You need to ensure that function should terminate gracefully if there are any I/O errors

Solution on next page

There can be multiple solutions but only one is provided for understanding purpose.

```
def user_account(filename):
    output_data = ""
    try:
        fread = open(filename,"r")
        headings = fread.readline().split(",")
    except FileNotFoundError: #fine if error is not mentioned
        print("Unable to open or read the file. Check filename or path and permissions to the
file.")
        return

    try:
        fname_idx = headings.index("First name")
        lname_idx = headings.index("Last name")
        dob_idx   = headings.index("Date of birth")
    except ValueError: #fine if error is not mentioned
        return

    for line in fread:
        linedata = line.split(",")
        login = linedata[fname_idx] + "." + linedata[lname_idx]
        dob = linedata[dob_idx].replace("/","")
        password = linedata[lname_idx] + "$" + dob
        output_data += login + "," + password + "\n"

    fread.close()
    try:
        fwrite = open("logins.txt","w")
        fwrite.write(output_data)
        fwrite.close
    except:
        print("unable to create the file for writing")
    return
```

---END OF EXAMINATION PAPER--

There can be multiple solutions but only one is provided for understanding purpose.