

1) The following code is riddled with bugs: some are due to bad syntax and some are due to bad logic. There are 7 bugs and each bug is worth one mark. **Only circle the actual bug and only circle at most 7 of them. One mark will be deducted for each additional item circled.** This question is scored out of 6 and a maximum of 7 will be awarded.

```
def readFromFile(fileName) :
    fileLines = ''
    try :
        inFile = fileName.open('r')
        # Read the entire file and store as a single string
        fileLines = inFile.read()
        inFile.close()
    except IOError:
        print(message)
        fileLines = None

    return inFile

def isWordInFile(word, fileName) :
    if not isinstance(word, int) :
        throw ValueError("word is not a string")

    contentsStr = readFromFile(fileName)
    if (contentsStr == None) :
        return False

    # Create a list of the words in the file separated by spaces
    contentsList = contentsStr.split('\t')
    return word in contentsList
```

2) Examine the following small Python programs. Determine the values of the variables at the end of each program's execution. There are 9 variables and each is worth one mark. This question is scored out of 8; a maximum of 9 will be awarded.

```
x = 1
y = 2
z = 3
```

```
def add(n, m) :
    global z
    x = 4
    y = 5
    z = n + m
    return z
```

```
y = add(y, x - 1)
```

x: _____ 1

y: _____ 2

z: _____ 2

```
def mult(a, b=0, c=1) :
    return (a + b) * c
```

```
d = mult(3)
```

```
e = mult(3, 2)
```

```
f = mult(b=3, a=3)
```

d: _____ 3

e: _____ 5

f: _____ 6

```
test = {'Q1': 'programming', \
        'Q2': 'output', \
        'Q3': 'programming'}
test['Q1'] = 'bugs'
del test['Q3']
testKeys = list(test.keys())
isAl = test['Q2'].isalpha()
```

test['Q1']: _____ 'bugs'

testKeys¹: _____ ['Q1', 'Q2']

isAl: _____ True

¹ The order of the elements in testKeys will not be considered

3. a) Write a function `listContents(...)` that has one string parameter which contains a complete directory path. The function must examine the contents of the given directory and generate two lists: one containing all the directories and one containing everything else. The function must then return the two lists. If an `OSError` is raised, the function must return two `None` values instead of the lists.

The import statements for the `os` and `os.path` modules are provided. Do not write a `main()` function or a call to `listContents(...)`. This question is scored out of 5.

```
import os
import os.path

def listContents(directoryPath) :
    dirList = []
    otherList = []
    try :
        fullList = os.listdir(directoryPath)
        for entry in fullList :
            fullPath = directoryPath + os.path.sep + entry
            if os.path.isdir(fullPath) :
                dirList.append(fullPath)
            else :
                otherList.append(fullPath)
    except OSError :
        return None, None

    return dirList, otherList
```

b) Write a function `saveContents(...)` with two string parameters. The first contains a complete directory path and the second contains the name of a text file with a default value of `"contents.txt"`. The function must call `listContents(...)` with the first parameter and capture the two lists returned. The contents of the directory list must then be written line-by-line in a text file. The file must be named using the second parameter. If `None` is returned by `listContents(...)` or if an `IOError` is raised then the function must return immediately.

Do not write a `main()` function or a call to `saveContents(...)`. This question is scored out of 5.

```
def saveContents(directoryPath, filename="contents.txt") :
    dirs, others = listContents(directoryPath)
    if (dirs == None) :
        return

    try :
        outFile = open(filename, 'w')
        for dir in dirs :
            outFile.write(dir + "\n")
        outFile.close()
    except IOError as message:
        return
```