**Solution to Semester 1 Examinations, 2017**

June 2017

1. Write a Python function `splitEmailAddress(Address)`, which, given an email address, e.g. "`Michael.Wise@uwa.edu.au`", returns a list of the component strings, e.g. [`'Michael', 'Wise', 'uwa', 'edu' 'au'`]. **[5 Marks]**

```
def splitEmailAddress(Address) :
    fields = []
    for f in Address.split('@') :
        for d in f.split('.') :
            fields.append(d)
    return(fields)
```

2. Write a Python definition for the function `basename(P)` which, given a pathname to a file (a string), returns a string with the file name without the preceding directories and without any suffix, if one exists. For example `basename('/Users/michaelw/CITS1401/exam.doc')` will return the string `'exam'`. (There is a function called `basename` in the `os.path` library, but please ignore it and instead use string processing functions.) **[5 Marks]**

```
# two possible ways it can be done
def basename(P) :
    return(P.split('/')[-1].split('.')[0])

def basename(P) :
    i = P.rfind('/')
    P = P[i+1:].split('.')[0]
    return(P)
```

3. The Manhattan distance between two points *(x1, x2)* and *(y1, y2)* – the distance a car needs to travel between two points in a city on a grid – is (in 2 dimensions):

   $$manhat = |(y_1 - x_1)| + |(y_2 - x_2)|$$

   Write a definition for `manhat(x, y)`, where `x` and `y` are points in N-dimensional space, each represented as a list of floating point values, e.g. `manhat([1, 3, 5, 7], [1, 9, 25, 42])` returns `41`. You can assume that the lists are the same length, though not necessarily length 4. (|..| stands for the absolute value function.) **[10 Marks]**

```
def manhat(x, y) :
    dist = 0
    for i in range(len(x)) :
        dist += abs(y[i]-x[i])
    return(dist)
```

4. Consider the following rather impenetrable (but correct) Python code, taken from a function:

```
if os.path.exists(f):
   return [c for c in open(f,'r').read().split('\n') if c != ""]
```

   a. What does the code do? **[3 marks]**
   b. Rewrite the code so that it is easier to understand **[7 Marks]**.

a.

The code:

- Tests whether file f exists
- Opens the file for reading
- Reads the file
- Splits the resulting string into lines (ie split on \n)
- Returns a those lines that are not blank

b.

```
# I've turned the code into a function for convenience. Not relevant to exam
# question.  Use of read also okay, but small marks loss (less efficient).
# use of strip() can be substituted. Important to separate out blank line test
def read_non_blank_lines(f) :
    if os.path.exists(f):
        infile = open(f, 'r')
        non_blanks = []
        for line in infile :
            line = line.strip()  # line = line[:-1] also fine
            if line != "" :
                non_blanks.append(line)
        return(non_blanks)
    return([])
```

5. What is seen as the result of executing the following Python code **[10 Marks]**:

```
def ft6b():
    numberGames = {}
    numberGames[(1,2,4)] = 5
    numberGames[(4,2,1)] = 10
    numberGames[(1,2)] = 12

    try:
        sum = 1
        for k in numberGames:
            sum *= numberGames[k]
        numberGames.append(sum)
        print('Try block executed.')
    except:
        print('Exception occured')
    print(numberGames, 'Sum=',sum)
```

Exception occured
{(1, 2): 12, (4, 2, 1): 10, (1, 2, 4): 5} Sum= 600

6. Write a definition for the function `merge(list1, list2)`, that, given two lists: `list1` and `list2`, which are sorted in ascending order, returns a list that combines the two lists in ascending order, e.g. `merge([1,3,5,11,12], [2,4,6,8])` returns `[1,2,3,4,5,6,8,11,12]`. (Hint: For starters, you will need a while loop that compares the smallest item in each list.) **[30 Marks]**

```
def merge(list1, list2) :
    mergedlist = []
    while list1 != [] and list2 != [] :
        if list1[0] <= list2[0] :
            mergedlist.append(list1[0])
            list1 = list1[1:]
        else:
            mergedlist.append(list2[0])
            list2 = list2[1:]
    if list1 != [] :
        mergedlist = mergedlist + list1
    else:
        mergedlist = mergedlist + list2
    return(mergedlist)
```

7. Write a definition for the function, `marksdistribution(D)`. The input to `marksdistribution` is a dictionary mapping student names to marks in the range 0..100. The output from `marksdistribution` should be a dictionary mapping marks ranges seen at UWA, to counts of students from `D` with marks in the respective ranges. The definitions of the marks ranges are:
    - N < 50
    - P ≥ 50 and < 60
    - Cr ≥ 60 and < 70
    - D ≥ 70 and < 80
    - HD ≥ 80

    For example, if `D = {"Fred":55, "James":67, "Jemima";71}`, `marksdistribution(D)` will return a dictionary resembling `{"P":1, "Cr":1, "D":1}` **[30 Marks]**

```
def marksdistribution(D) :
    grade_counts = {}
    for grade in ['N', 'P', 'Cr', 'D', 'HD'] :
        grade_counts[grade] = 0
    for mark in D.values() :
        if mark < 60 :
            if mark < 50 :
                grade_counts['N'] += 1
            else:
                grade_counts['P'] += 1
        elif mark >= 70 :
            if mark < 80 :
                grade_counts['D'] += 1
            else:
                grade_counts['HD'] += 1
        else:
            grade_counts["Cr"] += 1
    return(grade_counts)
```

8. Define a definition for the function, `pow(x, N)`, to compute $x^N$ for integer $x$ and integer $N$, e.g. $3^{1001}$. (Large numbers will require *long* integers.) For large $N$, a recursive function can be more efficient than multiplying $x$ $N$ times, so a recursive function will be awarded more marks. Specifically, if you create a function that uses repeated multiplication it will be awarded a maximum of **[10 marks]**. However, if you write a recursive function, your solution will be marked out of **[20 marks]**. Hint (for recursive solution): What happens if you divide $N$ by 2, i.e. first solve `pow(x, N//2)`.

```
def pow(x, n):
    if n  == 0 :
        return(1)
    if n == 1 :
        return(x)
    if n % 2 == 0 :
        x1 = pow(x, n//2)
        return(x1*x1)
    x1 = pow(x, n//2)
    return(x * x1 *x1)
```

**---END OF EXAMINATION PAPER--**