Part 1: Code Interpretation

Question 1

[10 marks]

Evaluate the following expressions, and provide the output in each case.

(a) `3 / 1`

A: *3.0*

(b) `'executrix'[3:6] + 'zippiest'[-3]`

A: *'cute'*

(c) `'completely different' and not('to be seen')`

A: *False*

(d) `sorted({'human': ['arthur', 'trillian'], 'robot': ['marvin']}.values())[1][-1]`

A: *'marvin'*

(e) `[str(i) + j for i in range(8, 10) for j in 'HD']`

A: *['8H', '8D', '9H', '9D']*

Question 2

[6 marks]

Rewrite the following function, replacing the `for` loop with a `while` loop, but preserving the remainder of the original code structure:

```
def same_colour(suits):
    SUIT_COLOURS = {'S': 1, 'H': 2, 'D': 2, 'C': 1}
    colour = ''
    for suit in suits:
        if not colour:
            colour = SUIT_COLOURS[suit]
        elif SUIT_COLOURS[suit] != colour:
            return False
    return True
```

A:

```
def same_colour(suits):
    SUIT_COLOURS = {'S': 1, 'H': 2, 'D': 2, 'C': 1}
    colour = ''
    i = 0
    while i < len(suits):
        suit = suits[i]
        if not colour:
            colour = SUIT_COLOURS[suit]
        elif SUIT_COLOURS[suit] != colour:
            return False
        i += 1
    return True
```

OR

```
def same_colour(suits):
    SUIT_COLOURS = {'S': 1, 'H': 2, 'D': 2, 'C': 1}
    colour = ''
    while suits:
        suit = suits.pop()
        if not colour:
            colour = SUIT_COLOURS[suit]
        elif SUIT_COLOURS[suit] != colour:
            return False
    return True
```

Question 3

[8 marks]

The function `animalise` is intended to take a string, identify each occurrence of a word which is also an animal name — as determined based on an input file with a single animal name per line — and replace each such word with `'animal'` (by default). For example, assuming the file `"animal-names.txt"` has the following content:

```
bear
bug
mouse
python
tick
worm
```

the function would produce the following output:

```
>>> orig = "bear with me as I fix the bug in my python code"
>>> animalise(orig, "animal-names.txt")
'animal with me as I fix the animal in my animal code'
```

As presented, the lines of the function are out of order. Put the line numbers in the correct order and introduce appropriate indentation (indent the line numbers to show how the corresponding lines would be indented in your code).

```
1  for line in f:
2  out += word + " "
3  adict = {}
4  out += newword + " "
5  out = ''
6  else:
7  if word in adict:
8  with open(afile) as f:
9  def animalise(text, afile, newword="animal"):
10 adict[line.strip()] = True
11 return out[:-1]
12 for word in text.split():
```

A: 9
3
8
1
10
5
12
7
4
6
2
11

Question 4

[9 marks]

The following function is meant to take a list `lst` and return the maximum value, based on recursion. The following are example outputs the code should produce:

```
>>> print(rec_max([]))
None
>>> print(rec_max([0, 4, -1, 2]))
4
>>> print(rec_max([0, 1, -1, 2, 0, 2]))
2
```

Identify exactly three (3) errors in the code (using the provided line numbers), determine for each whether it is a "syntax", "run-time" or "logic" error, and provide a replacement line which corrects the error.

```
1  def rec_max(lst):
2      n = len[lst]
3      if n < 1:
4          return None
5      if n > 1:
6          return lst[0]
7      max_rest = rec_max(lst)
8      elif lst[0] > max_rest:
9          return lst[0]
10     else:
11         return max_rest
```

A: *3 out of 4 of:*

- *line 2: run-time; should be* `len(lst)`
- *line 5: logic; should be* `n == 1`
- *line 7: logic; should be* `rec_max(lst[1:])`
- *line 8: syntax; should be* `if`

Question 5

[8 marks]

The following code is intended to validate whether the score for a single (possibly incomplete) set of tennis is valid or not, but contains logic errors:

```
def is_set(score_str):
    game_score = set_score = False
    for score in score_str.split(" "):
        try:
            s1, s2 = score.split("-")
            if (s1 and s2 in ('0', '15', '30', '40', 'Ad')) and not game_score:
                game_score = True
            elif 0 <= int(s1) <= 6 and 0 <= int(s2) <= 6:
                if game_score or set_score:
                    return False
                set_score = True
            else:
                return False
        except ValueError:
            return False
    return True
```

A valid set score is defined to be a string consisting of (in sequence):

- the game score for completed games in the set, in the form of two non-negative integers separated by a hyphen (`"-"`), where each game score is between 0 and 6 inclusive (e.g. `'4-5'` or `'0-6'`); in the case that no games have been completed, this component of the set score should be missing entirely.
- the point score of the current game (if the first point has been played, but the game has not been completed) in the form of two strings separated by a hyphen (`"-"`); the point score for each player should be one of the following values:

  `'0'`, `'15'`, `'30'`, `'40'`, `'Ad'`

  noting that `'Ad'` can only occur if the other player's score is `'40'`. Example game point scores are `'30-15'` and `'Ad-40'`. In the case that no points have been played in the current game, or the game has completed, no point score should be presented.

The game score and point score (assuming both are present) should be separated by a single space (e.g. `'3-4 40-15'`).

Note that for the purposes of this question, we define a set to not have tiebreakers, and be won by the first player to win 6 games.

(continues on next page)

The provided code is imperfect, in that it sometimes correctly validates (or invalidates) set scores (as defined by the definition above), but equally, sometimes analyses a valid set score as invalid and sometimes analyses an invalid set score as valid.

(a) Provide an example of a valid set score that is correctly analysed as being valid by the provided code (i.e. a valid input where the return value is `True`):

A:

(b) Provide an example of an invalid set score that is correctly analysed as being invalid by the provided code (i.e. an invalid input where the return value is `False`):

A:

(c) Provide an example of an invalid set score that is *in*correctly analysed as being valid by the provided code (i.e. an invalid input where the return value is `True`):

A: `'Ad-Ad'`, `'0-Ad'`, `'0-0'`, `'6-15'`, *etc.*

(d) Provide an example of a valid set score that is *in*correctly analysed as being invalid by the provided code (i.e. a valid input where the return value is `False`):

A: `"5-0 15-0"` *etc. (game score for player 1 is non-zero, with valid point score)*

Part 2: Generating Code

Question 6

[10 marks]

Write a single Python statement that generates each of the following exceptions + error messages, assuming that it is executed in isolation of any other code.

(a) `ValueError: invalid literal for int() with base 10: 'one'`

A: `int('one')`

(b) `ZeroDivisionError: division by zero`

A: *`x/0` where `x` is any number*

(c) `NameError: name 'y'is not defined`

A: *`y += 1`, `x = y`, etc.*

(d) `TypeError: unhashable type: 'list'`

A: *`{ [] }` or any other set or dictionary with a list as a key*

(e) `IndexError: list index out of range`

A: *`[][1]` etc.*

Question 7

[10 marks]

Binary search is a divide-and-conquer approach for locating a given value in a sorted list, and returning its position in the list (or `None` in the case that the value is not contained in the list). Complete the following iterative implementation of binary search by providing a single statement to insert into each of the numbered boxes. Note that your code should run at the indentation level indicated for each box.

```
def search(val, nlist):
    [ 1 ]
    end = len(nlist) - 1
    while start < end:
        [ 2 ]
        if nlist[mid] == val:
            [ 3 ]
        elif nlist[mid] < val:
            start = mid + 1
        else:
            [ 4 ]
    [ 5 ]
```

(1) **A:** *start = 0*

(2) **A:** *mid = (start + end)//2*

(3) **A:** *return mid*

(4) **A:** *end = mid*

(5) **A:** *return None*

Question 8

[19 marks]

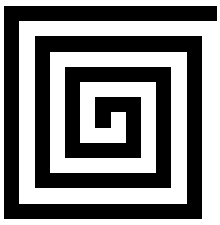
Write a function `spiral(size, direction, fname)` that generates a black spiral emanating from the centre of an RGB image with white background, of dimensions `size`×`size` pixels where `size` is a positive, odd integer. Each edge of the spiral should be of width one pixel, and strictly horizontal or vertical, with parallel adjacent edges separated by one pixel. The starting direction of the spiral from the centre of the image is defined by `direction` as follows:

- 0: start upwards
- 1: start to the right
- 2: start downwards
- 3: start to the left

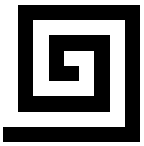
In each case, the spiral should move one pixel away from the centre of the image, and spiral clockwise (with the central pixel always being black). The third argument, `fname`, is the name of the file that the image should be saved to.
For example, given the following function calls:

```
>>> spiral(15, 0, "spiral15.png")
>>> spiral(9, 3, "spiral9.png")
```

The two generated images would be as follows:



spiral15.png



spiral9.png

To save the image, you should make use of the following function in your function definition:

```
def save_img(dim, pxl, fname, mode='RGB'):
    """save an image of dimensions `dim` (a 2-tuple),
    made up of the pixels contained in `pxl` (a dictionary
    with x,y coordinates as keys, and RGB 3-tuples as values)
    in file `fname`, based on image mode `mode`"""
    img_out = Image.new(mode, dim)
    pix_out = img_out.load()
    for x in range(dim[0]):
        for y in range(dim[1]):
            pix_out[x, y] = pxl[x, y]
    img_out.save(fname)
```

A:

```
import itertools

def spiral(size, direction, fname):
    DIR = itertools.cycle([(-1, 1), (1, 0), (1, 1), (-1, 0)])
    for i in range(direction):
        next(DIR)
    pixels = {}
    for x in range(size):
        for y in range(size):
            pixels[x, y] = (255, 255, 255)
    half_size = size//2
    start = [half_size, half_size]
    step = 1
    for i in range(size):
        direction, xy = next(DIR)
        for j in range(step):
            pixels[tuple(start)] = (0, 0, 0)
            start[xy] += direction
        step += 1
    save_img((size, size), pixels, fname)
```

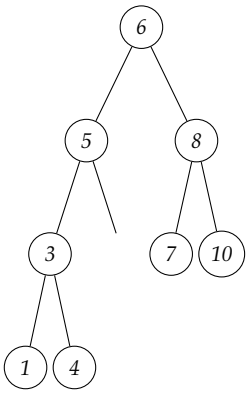
Part 3: Conceptual Questions and Applications of Computing

Question 9: Data Representation

(a) Given the following list of numbers `[5, 8, 3, 4, 7, 10, 1]` and an initial tree that contains the single value `6`, draw the binary search tree in the box below that results when these numbers are inserted in sequence.

[6 marks]

A:



(b) Each of the following represents a greyscale value, in the form of an 8-digit binary number. Provide the hexadecimal equivalent for each of these values.

[6 marks]

(1) 10101010

A: *AA*

(2) 01101011

A: *6B*

(3) 10010011

A: *93*

Question 10: Applications of Computing

(a) With the aid of an example, describe what an API (Application Programming Interface) is.

[3 marks]

A: *a set of functions and procedures that allow the creation of applications which access the feature or data of an operating system, application, or other service*

(b) In the context of the internet, what is the purpose of a digital certificate?

[3 marks]

A: *an electronic document used to prove the ownership of a public key.*

[4 marks]

Question 11: URLs and the Web

With reference to the following URL:

```
ftp://ftp.downloads.com:21/spam/spam
```

provide the portion of the URL which stipulates each of the following:

(i) the path

A: *spam/spam*

(ii) the host name

A: *ftp.downloads.com*

(iii) the port

A: *21*

(iv) the protocol

A: *ftp*