```
# Q1
# one-liners

#'suboptimal'[2:][:3] + '!'
# 'bop!'

'snout'[2:]
# 'out'

sorted(['fog', 'cog', 'dog'])
# ['cog', 'dog', 'fog']

#('hurry'[-1] + 'south'[-1] + 'now'[-1])[::-1]
('now'[-1] + 'south'[-1] + 'hurry'[-1])
# 'why'

#sorted({'owls': 'good', 'more owls': 'not so
good'}.values())[1].split()[0]
sorted({'owls': 'good', 'more owls': 'not so good'}.values())[1]
# 'not so good'

'nic' not in sorted('cinema')
# True

[i%2 for i in range(0, 10, 2)]
# [0, 0, 0, 0, 0]



# Q2
# convert from for loop to while loop
# function to check if a word contains one instance of all five vowels

# for loop
def all_vowels(word):
    vowels = 'aeiou'
    seen = ''
    for c in word:
        if c in vowels:
            seen += c
    return sorted(seen) == list(vowels)

# while loop
def all_vowels_w(word):
    vowels = 'aeiou'
    seen = ''
    i = 0
    while i < len(word):
        if word[i] in vowels:
            seen += word[i]
        i += 1
    return sorted(seen) == list(vowels)
```

```
# Q3
# find errors

# In project 1, you considered preferential voting, in which a valid
vote
# required a voter to list all candidates once in there preferred
order.
#
# The following function is intended to recursively generate all
possible valid
# votes for a given set of n candidates.

def orderings(candidates):
    if not candidates:
        return [candidates]

    # vote == []  ## runtime error
    vote = []
    for i in range(len(candidates)):
        # current = candidates(i)   ## runtime error
        current = candidates[i]
        # remainining = candidates[i] + candidates[i+1:]  ## runtime
error
        remaining = candidates[:i] + candidates[i+1:]
        # for o in orderings(remaining)  ## syntax error
        for o in orderings(remaining):
            # vote.append([candidates] + o)  ## logic error
            vote.append([current] + o)

    return vote
```

```
# Q4
# shuffled code (dictionary provided unshuffled)


def obfuscate_text(text):
    short_text = ''
    prev_char = ''
    for c in text:
        if c == prev_char:
            continue
        short_text += c
        prev_char = c
    obs_text = ''
    i = 0
    while i < len(short_text):
        if short_text[i] in subs:
            obs_text += subs[short_text[i]]
        elif i%2 == 0:
            obs_text += short_text[i].upper()
        else:
            obs_text += short_text[i]
        i += 1
    return obs_text
```

```
### blank following line ###
import csv

def add_proportions(csv_filename, new_filename):

    with open(csv_filename) as csv_file:
        reader = csv.DictReader(csv_file, skipinitialspace=True)

        header = reader.fieldnames
        data = list(reader)

    count_sum = 0
    ### blank following line ###
    for row in data:
        count_sum += int(row['count'])

    for row in data:
        prop = int(row['count']) / count_sum * 100
        row['proportion'] = round(prop, 2)

    ### blank following line ###
    new_header = header + ['proportions']

    ### blank following line ###
    with open(new_filename, 'w') as new_file:
        writer = csv.DictWriter(new_file, new_header)

        writer.writeheader()

        ### blank following line ###
        for row in data:
            writer.writerow(row)
```

```
# Q6
# generate errors

# write a single Python statement that generates each of the following
# errors and exceptions, assuming it is executed in isolation of any
other
# code.

# IndexError: list index out of range

[][0]

# TypeError: 'tuple' object does not support item assignment

(0,)[0] = 0

# AttributeError: 'str' object has no attribute 'len'

'0'.len()

# KeyError: 0

{}[0]

# TypeError: unsupported operand type(s) for +: 'int' and 'str'

1 + "1"
```

```
# Q7
# coding question



def create_string(str1, str2, start, end):
    result = ''
    i = 0

    for j in range(start, end):
        result = result + str1[i] + str2[j]
        i += 1

    return result


def alternate(word):
    vlist = []
    clist = []
    for c in word:
        if c in 'aeiou':
            vlist += c
        else:
            clist += c

    nc = len(clist)
    nv = len(vlist)

    if abs(nc - nv) > 1:
        return None

    if nc < nv:
        return vlist[0] + create_string(clist, vlist, 1, nv)
    elif nv < nc:
        return clist[0] + create_string(vlist, clist, 1, nc)
    elif clist[0] < vlist[0]:
        return create_string(clist, vlist, 0, len(clist))
    else:
        return create_string(vlist, clist, 0, len(vlist))
```

Q7

Conceptual questions

(a) calculates a solution with a guarantee of correctness

(b) yes, as it divides the remaining list of items in half at each iteration

Q9

Applications of computing questions

(a) AI: developing computer systems that can perform tasks that traditionally can only be done by a human

(b)

Learn a model of “normal” traffic

Use this model to test new traffic for anomalies

Any anomalies are treated as an attack

Q10

HTML question

```
<!(*@\blank[1]@*) html>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<(*@\blank[2]@*) http-equiv="Content-Type" content="text/html;
charset=UTF-8">

<title>COMP10001: The Final Exam</title>

</head>

<(*\blank[3]@*)>

<h1>COMP10001: The Final Exam</h1>

<p>Starring:

<(*@\blank[3]@*)>

  <li>You!</li>

  <li>Chris, Nic, Marion</li>

  <li><(a (*@\blank[4]@*)="./images/surprised_pikachu.gif"
alt="Pikachu!"/></li>

</ul>

</p>

</body>

</(*@\blank[5]@*)>
```

DOCTYPE

meta

ul

src

html