

- If a multiple choice question has several correct answers, you must give all correct answers to receive credit.
 - **Write clearly.** Answers that are difficult to read may receive 0 points.
 - You must pass part A (4 correct out of 8 questions) to have your part B graded.
 - Write only on one side of each paper!
 - No **import** are allowed in your answers in part B (neither Python's standard modules nor external libraries) unless it is explicitly mentioned in the assignment. Built-in functions like **len**, **range**, **print**, and **sum** are allowed.
 - **Aids:** An A4 with as much information as you want. You can write on both sides.
 - **Grade thresholds:** E: 10, D: 12, C: 14, B: 16, A: 18, of maximum 20.
-

Part A: Multiple choice

Please collect your answers to part A on a single piece of paper.

1. What is printed by the code to the right?

- A. -2
- B. 0
- C. 6
- D. 22
- E. An exception is raised.

```
y = 10
for x in [-5, -4, -3]:
    y += x
print(y)
```

2. What will be the result of the code to the right?

- A. DA7066
- B. 7.5
- C. MM2001
- D. 30
- E. An error message

```
dct = {'DA7066': 7.5, 'MM2001': 30}
print(dct[1])
```

3. Which two errors can occur when using the function huh below where dct is a dictionary?

- A. KeyError
- B. ValueError
- C. ZeroDivisionError
- D. SyntaxError
- E. IndexError

```
def huh(dct):
    s = 0
    for x in dct.keys():
        y = dct[x]
        s += float(x)/float(y)
    return s
```

4. Which of the following statements are true for Unix (assuming the bash interpreter as used in the course compendium)?

- A. The `|` command is used to transfer data between commands (pipelines)
- B. `rm file.txt` removes the file `file.txt`
- C. `mkdir` can be used to create a folder.
- D. `wc` can count lines, characters, and words in files.
- E. `head` displays the beginning of a file.

5. What is the result of the code to the right?

- A. -17
- B. -3
- C. -2
- D. 0
- E. None
- F. An error message.

```
def max(lst):
    val = 0
    for elem in lst:
        if elem > val:
            val = elem
    return val
print(max([-3, -2, -17]))
```

6. How many times is 'I am in f' printed in the code below?

- A. 0
- B. 1
- C. 2
- D. 3
- E. Infinitely many times

```
def f(x):
    if x > 0:
        print('I am in f')
        g(x-1)
    else:
        print('Bye')

def g(x):
    if x > 0:
        print('I am in g')
        f(x-2)
    else:
        print('Bye')

print(f(4))
```

7. What will be the value of the variable `text` after running the code to the right?

- A. "" (an empty string)
- B. 'ABC'
- C. 'ACE'
- D. 'ACF'
- E. 'ACEG'

```
text = ""
msg= "ABCDEFGH"
i = 0
while len(text) < 3:
    text += msg[i]
    i += 1
msg = msg[i:]
```

8. What will be the contents of `s` after the code to the right is run?

- A. `ixdxx`
- B. `xnxex`
- C. `ixned`
- D. `ixixi`
- E. `xixix`

```
lis = ['i', 'n', 'd', 'e', 'x']
s = ''
for i in range(len(lis)):
    if i % 2 == 0:
        s += lis[i]
    else:
        s += lis[-1]
```

Part B: Coding

Please use a separate piece of paper (or several) for each question in part B. Multipart questions such as 9A and 9B can be written on the same piece of paper.

9.

- A. Write a function `first_n_prod` that computes the product of the first `n` integers in a list. For example, `n=3` and `[1,2,3,4,5]` results in $1 * 2 * 3 = 6$. If there are fewer than `n` integers in the list it should return 0. You can assume that input have correct datatypes. (2p)

Example use:

```
[In: ] print(first_n_prod(1, [1,2,3,4,5]))
[Out:] 1
[In: ] print(first_n_prod(7, [1,2,3,4,5]))
[Out:] 0
[In: ] print(first_n_prod(3, []))
[Out:] 0
[In: ] print(first_n_prod(3, [1, 2, -4]))
[Out:] -8
```

- B. Extend your function `first_n_prod` such that if it finds an element that is not of the type `int` it prints 'Illegal input' and returns a 0. (1p)

Tip: You can use the built-in function `type` that returns the type of the variable.

Example usage:

```
[In: ] print(first_n_prod(1, [1,'2',3,4,5]))
[Out:] Illegal input
[Out:] 0
[In: ] print(first_n_prod(7, [1,2,True,4,5]))
[Out:] Illegal input
[Out:] 0
[In: ] print(first_n_prod(3, [(1,2,3), 2]))
[Out:] Illegal input
[Out:] 0
```

10. The hacker @Kim_L337_Haxor has written a script to parse out TODO comments from Python files that we can assume are always written as `#TODO` followed by the text of the task). See the code below. The code contains precisely two errors (can be either syntactical or semantical). What should be corrected for the code to work as intended? One point per found error. (2p)

```
def todo_finder(l):
    for i in len(range(l)):
        if l[i] == '#TODO':
            return l[i:]
    return ''

def get_all_todos(in_filename, out_filename):
    with open(in_filename, 'r') as infile, open(out_filename, 'w') as outfile:
        for line in infile:
            todo = todo_finder(line)
            if todo:
                outfile.write(todo)
```

11. The collatz sequence c_0, c_1, c_3, \dots is a sequence of positive integers. It is generated by starting with any positive number `n` and following just two simple rules: If `n` is even, divide it by two `n/2`, and if it's odd, triple it and add one $3*n+1$. Write a function `collatz(n)` that returns the collatz sequence up until and including the first 1 that appear in the sequence. (2p)

Example usage:

```

[In: ] print(collatz(1))
[Out:] [1]
[In: ] print(collatz(3))
[Out:] [3, 10, 5, 16, 8, 4, 2, 1]
[In: ] print(collatz(2**5))
[Out:] [32, 16, 8, 4, 2, 1]

```

12. Write a function `flip_dict` that takes a dictionary `dict` and returns a new dictionary where keys and values have swapped places. The function only needs to handle simple dictionaries like the ones in the examples below, and it is assumed that no two keys are associated with the same value in the dict. (2p)

Example usage:

```

[In: ] flip_dict({'DA7066' : 7.5})
[Out:] {7.5 : 'DA7066'}
[In: ] flip_dict({1 : 99, 'a': 2, 'c': True})
[Out:] {99: 1, 2: 'a', True: 'c'}
[In: ] flip_dict({})
[Out:] {}

```

13. Give an example of a dictionary where your `flip_dict` function would give a runtime error. You can only get a point here if the `flip_dict` function you wrote actually works for the examples above. (1p)

14. Write a function `add_matrices` that takes two $N \times M$ matrices A and B as input and perform element wise addition and returns the resulting matrix C . With element-wise addition we mean $c_{ij} = a_{ij} + b_{ij}$, $i \in [0, N - 1], j \in [0, M - 1]$ (see an example below). You should decide yourself the suitable data representation for a matrix. However, the resulting matrix (after addition) **has to be** of the same data structure as the input matrices. You can assume correct input to the function such as the matrices being of the same (non-zero) dimensions and all elements are of the same data types, i.e., no error handling is needed. (2p)

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} + \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} a_{00} + b_{00} & a_{01} + b_{01} \\ a_{10} + b_{10} & a_{11} + b_{11} \end{pmatrix}$$

An example with numbers

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 9 \\ 3 & 71 \end{pmatrix} = \begin{pmatrix} 2 & 11 \\ 6 & 75 \end{pmatrix}$$