CS1010X — Programming Methodology
National University of Singapore

Practical Examination

Time allowed: 2 hours

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to refer to any hard copy material and not electronic materials.
2. You are to do your work without any assistance from another intelligent human being – if found otherwise, you will receive **ZERO** for the practical exam and will be subject to other disciplinary actions.
3. This practical exam consists of **3** "topics" printed in **5** pages (inclusive of this cover page).
4. The maximum score of this quiz is **100 marks** and you will need to answer all questions to achieve the maximum score. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question. You are advised to use your time wisely and do move on to other parts if one part is difficult and can take up lots of time to solve.
5. Your answers should be submitted on Coursemology.org **BEFORE** the end of the exam. If you have any submissions timestamped with a time after the exam has ended, your submission for that question will not be graded. Remember to **finalize** your submissions before the end of the exam.
6. You will be allowed to run some public tests cases to verify that your submission is correct. Note that you can run the test cases on Coursemology.org up to a **maximum of 10 times** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness using IDLE and not on Coursemology.org. Do however ensure that you submit your answers correctly by running the test cases at least once.
7. You are also provided with the template practical-template.py to work with. If Coursemology.org fails, you need to rename your file practical-<mat no>.py where <mat no> is your matriculation number and submit that file by leaving it in your computer.
8. Please note that while sample executions are given in some cases, it is not sufficient to write programs that simply satisfy the given examples. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get the allocated credit.
9. Please behave like a good programmer to make your codes readable with good naming convention for the variables and function names. If you do not do so, we reserve the right to deduct small credit even if your program is correct.

10. **To ensure that you do the proper analysis and design before you code the solutions, you are not allowed to touch the computer for the first 30 minutes. The TA will let you know when you are allowed to do so.**

GOOD LUCK!

Q1. [10 marks] Given a string of lower case alphabets, count the occurrences of the characters in the given string. Obviously, characters that do not appear in the given string will have zero occurrence. Then sort the characters in descending order of the occurrences. If there are ties, the lower characters come first. This output string will be used by Q2 and Q3 to do encryption and decryption.

For example, given the string “the lazy fox jumps over a brown dog”, the output is

“oaerbdfghjlmnpstuvwxyzcikq”

o	a	e	r	b	d	f	g	h	j	l	m	n	p	s	t	u	v	w	x	y	z	c	i	k	q
4	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

You should use the following function heading:

```
def generate_key(s):
```

where s is the given string of lower case alphabets.

Note: you may use the built in sort in your solution or use your own sorting helper function.

Q2. [10 marks] Given a message, encrypt the message using the output from Q1 as the encryption key. Characters that are not lower case alphabets remain the same in the encrypted output.

For example, “##Hello World!!” will be encrypted as “##Hbmms Wsvmr!!”

o	a	e	r	b	d	f	g	h	j	l	m	n	p	s	t	u	v	w	x	y	z	c	i	k	q
4	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

You should use the following heading:

```
def encrypt(msg, s):
```

where msg is the message to be encrypted, and s will be used to generate the key by the function in Q1.

Q3. [10 marks] Given an encrypted message, decrypt the message using the output from Q1 as the encryption key. Characters that are not lower case alphabets remain the same in the decrypted output.

For example, “##Hbmms Wsvmr!!” will be decrypted as “##Hello World!!”

You should use the following heading:

```
def decrypt(msg, s):
```

where msg is the message to be encrypted, and s will be used to generate the key by the function in Q1.

Note: For Q2 and Q3, you can assume that the function generate_key(s) exists.

Q4. [10 marks] Given an array A with length n, find the "leader" for each element A[i]. The leader of A[i] is defined as the nearest element to its right that is larger than A[i]. For example, if the array is:

A = [1, 2, 1, 3, 2, 4, 5, 7, 6]

The leader of A[0] is A[1] = 2, the leader of A[1] is A[3] = 3, the leader of A[2] is also A[3] = 3, so on and so forth. If there is no element on the right that is larger than A[i], then A[i] is its own leader. Please write a function *leader(A)* that outputs a list, where the i-th element of the list is the leader of A[i].

For example, leader([1, 2, 1, 3, 2, 4, 5, 7, 6]) should output [2, 3, 3, 4, 4, 5, 7, 7, 6].

Note that n could be very large, and a solution with time complexity $O(n^2)$ cannot pass the evaluation test cases.

Q5. [20 marks] Given an array A with length n, this time we want to find the "k-leader" for each A[i]. The k-leader of A[i] is the nearest element to its right whose distance to position i is at least k and value is larger than A[i]. If there is no element satisfying these conditions, A[i] is its own leader. Please write a function *k_leader(A, k)* that outputs a list of k-leaders.

For example, k_leader([1, 2, 1, 3, 2, 4, 5, 7, 6], 2)
should output [3, 3, 2, 4, 5, 7, 6, 7, 6].

Note that n could be very large, and a solution with time complexity $O(n^2)$ cannot pass the evaluation test cases.

Q6. [10 marks] Given a *sorted* array A with length m, we want to find an integer number x in A that minimizes the summation, $\sum_i |A[i] - x|$. That is, the sum of the absolute values of the differences between x and each A[i] is minimum.

Please write a function that takes in the array A and outputs the smallest possible value of $\sum_i |A[i] - x|$.

You should use the following heading:

def minimum(A):

For example, given the sorted array

A = [1, 2, 3, 5, 6, 7], the output should be 12, and

A = [1, 3, 7, 8, 11, 17, 25, 31], the output should be 65

Note that m could be very large, and a solution with time complexity $O(m^2)$ cannot pass the evaluation test cases.

Q7. [10 marks]

Given an integer number k and *three* sorted arrays $A[0], A[1], A[2]$ stored in the 2D list A, whose entries are between 0 and 10, we want to find three integer numbers $x[0], x[1], x[2]$ satisfying $|x[1] - x[0]| \geq k$ and $|x[2] - x[1]| \geq k$ that minimizes $\sum_i \sum_j |A[i][j] - x[i]|$.

For example, given the nested list

A = [[1, 2, 3, 5, 6, 7], [2, 2, 3, 3, 6, 7], [1, 1, 3, 5, 5]] and k=2,
then $x[0] = 5, x[1] = 3, x[2] = 1$ minimizes $\sum_i \sum_j |A[i][j] - x[i]|$.
The smallest value is 31.

You should use the following heading:

```
def constrained_minimum(A, k):
```

Hint: for this question you may use brute force.

Q8. [20 marks]

Following Q7, this time we are given an integer number k and n sorted arrays $A[0], \dots, A[n-1]$ stored in the 2D list A, whose entries are still between 0 and 10, we want to find n integer numbers $x[0], \dots, x[n-1]$ satisfying $|x[i] - x[i-1]| \geq k$ for all $0 < i < n$ that minimizes $\sum_i \sum_j |A[i][j] - x[i]|$.

For example, given the nested list

A = [[1, 2, 3, 5, 6, 7], [2, 2, 3, 3, 6, 7], [1, 1, 3, 5, 5], [1, 2, 5, 8, 9]] and k =2,

then $x = [5, 3, 1, 5]$ minimizes the value, which is 45.

You should use the following heading:

```
def constrained_minimum_adv(A, k):
```

Note that n is relatively large (could be 100), and a brute force solution will not pass.

Hint: Observe that the constraint on x is only for consecutive positions. This means the value of $x[i]$ only depends on $x[i-1]$ and $A[i]$.