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!

Topic 1: Encoding/Decoding [30 marks]

Assume that we have a string S that we want to encode it, and of course, decode it subsequently.
Let's call each character in S as c, and let c_a be the ASCII code of c (which in Python can be obtained through "ord(c)"). As our encoded string will include the symbol "#", and we thus are guarantee that c is never equal to "#".

To encode, let b be a number between 3 and 9, and b is called the base. For each character c in S, we encode it as c' = "#" + str(b) + "#" + str(c_a converted to base b), and then concatenate together all the c' in the order of the characters c in S. See question 2 about how b is obtained.

To decode, convert each c_a back to decimal using the same base used (can be found in the #b# in front of each c_a, and concatenate all of them into a string. Note that chr(num) returns the character which has ASCII value num

==

Question 1. Write a helper function to convert a decimal integer to a number with the given base. You should use the given function heading
def dec_to_base(dec, base): **[5 marks]**

For example, calling dec_to_base(123, 5) returns 443
And calling dec_to_base(443, 9) returns 542

==

Question 2. Write a function to encode the given string S into a secret message as described above. The base used for converting decimal value to value in the base is given in a list of variable length, base_lst. That is, base_lst can be any number of n_i, where 3 <= n_i <= 9. You should then use the n_i in the base_lst in round robin manner. That is, to convert the first character in S, you should use base_lst[0], to convert the second character in S, use base_lst[1], and so on. When you reach the last base in base_lst, you go back to base_lst[0] again.

You should use the given function heading
def encode(S, base_lst): **[10 marks]**

For example, calling encode("hello",[5,3,7]) should produce
"#5#404#3#10202#7#213#5#413#3#11010#"

(Note: For this Question, the function dec_to_base() is provided in the Coursemology environment. When you write your solution for Q2, you can assume the function dec_to_base() exists and you can just call it in your solution.)

Question 3. Write a helper function to convert a number in a given base to decimal. You Should use the given function heading

def base_to_dec(num, base): [5 marks]

for example, calling base_to_dec(11010,3) should return 111

==

Question 4. **Write a function to decode the secret message as described above. You should use the following function heading where S is the secret message produced in question 2**

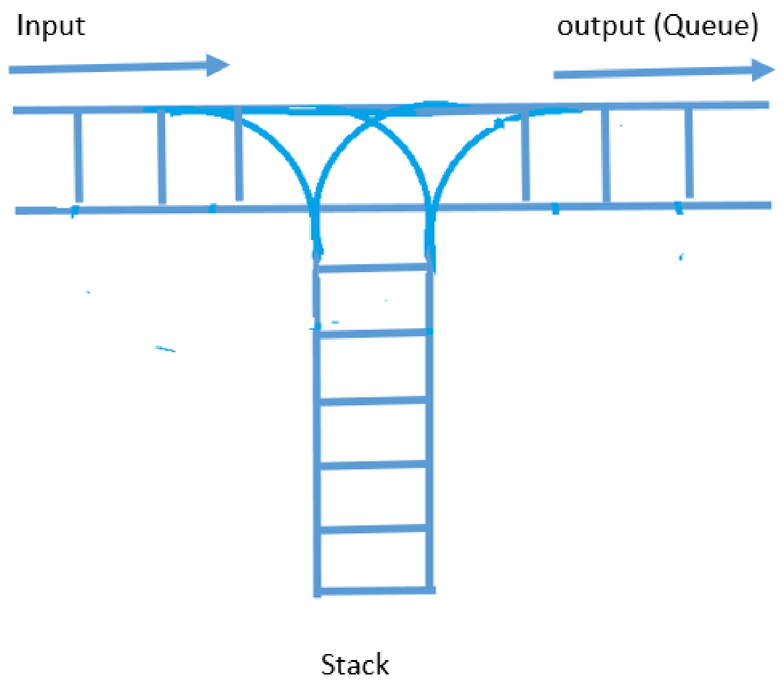
def decode(S): [10 marks]

For example, calling decode("#5#404#3#10202#7#213#5#413#3#11010#") should produce "hello"

(For Question 4, the function base_to_dec() is provided in the Coursemology environment. When you write your solution for Q4, you can assume the function base_to_dec() exists and you can just call it in your solution.)

Topic 2: Stack and Queue [30 marks]

The picture below is like a railway shunting yard. It is used by transportation/logistic companies to put train cargo cars in the correct order before they leave the train station. After starting the journey, when they reach a station, they just leave the cargo car(s) meant for that station and continue its journey as those cargo car(s) is/are at the end of the train. That is you can use the following system to arrange the cargo cars in the desired order



In this question, you are supposed to use a stack and a queue to simulate the operations in the shunting yard. The incoming cargo cars are ordered in increasing order of non-repeating but not necessarily consecutive numbers. For example, [3, 6, 7, 18, 21, 45, 66, 78, 214, ……..]. To obtain the desired order, you may send the cargo car from the input to the output (queue) or you can put it into the stack (push) and output (pop_and_queue) it later. So there are three possible commands when a cargo car arrive.

1. Queue (Q): send to output
2. Push (P): push into the stack
3. Pop_and_queue (PQ): pop from stack and send to output

For example, if the input sequence is [1, 4, 7, 9, 11, 15] and the commands are [P, Q, PQ, Q, P, P, Q, PQ, PQ], Then the output would be [9, 11, 15, 7, 1, 4]. 4 is the first cargo car in the train and 9 is the last cargo car in the train. The process is illustrated below. Use this illustration to help you understand the input, output and the operations.

Input	command	output	stack
1	P	[]	[1]
4	Q	[4]	[1]
	PQ	[1, 4]	[]
7	Q	[7, 1, 4]	[]
9	P	[7, 1, 4]	[9]
11	P	[7, 1, 4]	[9, 11]
15	Q	[15, 7, 1, 4]	[9, 11]
	PQ	[11, 15, 7, 1, 4]	[9]
	PQ	[9, 11, 15 7, 1, 4]	[]

Question 5: Write a function that accept a list of sorted integers representing the cargo cars and a list of commands and output the resulting order of the cargo cars. [10 marks]
You should use the following heading

```
def arrange(car_lst, commands):
```

Note: You do not need to implement the stack and queue ADT. Just simply use the relevant list operations such as append() to add an item to the end of the list and pop() to remove the last item from the list

Question 6: Write a function to accept two list of integers and determine if it is possible to produce the second list using a series of commands on the first list. Note that the first list is a sorted list without duplicate. Output "possible" or "impossible" accordingly.
You should use the following heading
def check(car_lst_in, car_lst_out): **[20 marks]**

For example:
car_lst_in = [1, 4, 7, 23, 45, 67], car_lst_out = [67, 45, 23, 7, 4, 1],
output = possible
c~~C~~ar_lst_in = [1, 2, 4, 6, 7, 9, 11, 15, 16], car_lst_out = [4, 6, 16, 7, 15, 9, 11, 2, 1]
output = possible
c~~C~~ar_lst_in = [1, 2, 4, 6, 7, 9, 11, 15, 16], car_lst_out = [4, 7, 16, 6, 15, 9, 11, 2, 1]
output = impossible.

Hint: smaller values should be in reversed order if larger values had come out first since we need to put the smaller values into the stack. Hence, 6 cannot be in front of 7 in the last example.
==

Topic 3: Complexity is important [40 marks]

Given a list of n integers, lst, where $3 <= n <= 10^5$, a ACB pattern is a subsequence of three integers lst[i], lst[j] and lst[k] such that lst[i] < lst[k] < lst[j] and i < j < k.
Return True if there is a ACB pattern in lst, otherwise return False,

Question 7: Write a function that implement a brute force algorithm to solve the problem.
You should use the given heading **(10 marks)**
def ACB(lst):
For example, lst = [1,2,3,4,5,6,7,8], output False
lst = [1,2,3,4,7,8,9,5], output True

Question 8: Write a function that implement a more efficient algorithm to solve the problem. Again, use the given heading **(up to 30 marks)**
def ACB(lst):
For example, lst = [1,2,3,4,5,6,7,8], output False
lst = [1,2,3,4,7,8,9,5], output True

(Note: For 2/3 of the test cases, the length n of the array is at most 2000, and for the rest of 1/3 of the test cases, n is 10^5. If you only pass the small test cases, you will get 20 marks, and if you pass all the test cases, you can get full marks. Most likely if you want to get full marks, your function should have a time complexity of O(n), and a solution with O(n^2) complexity can only get up to 20 marks. The mark is given based on your last submission.

== end of paper ==