دانشکده مهندسی کامپیوتر

درس ساختمان های داده

تمرین سری۲

مدرس ..... دکتر حسین رحمانی

طراحان ..... کامیار مرادیان زه آب – آیسا میاهی نیا

تاریخ انتشار ۱۴۰۱/۸/۹

تاریخ تحویل ۱۴۰۱/۸/۱۶

در رابطه با تمرین

- این تمرین شامل مباحث:
  - LinkedList
  - Stack
  - Queue
  - Data Types

می باشد.

- برای پاسخگویی به سوالات این تمرین نیاز است که به مطالب اسلاید های Data Types , LinkedList , Stack , Queue مسلط باشید.
- نمره ی این تمرین از ۱۰۰ می باشد و بارم هر سوال روبه‌روی آن نوشته شده است.
- این تمرین دارای ۱ سوال عملی می باشد که باید با استفاده از زبان سی شارپ به آنها داخل سامانه کوئرا پاسخ دهید.
- اگر برای جواب دادن به سوالی نیاز به پیش فرضی دارید، فرض خود را صریحاً نوشته و با توجه به فرض خود به ارائه جواب بپردازید.
- فایل پاسخ تئوری خود را به صورت hw#_student-id.pdf نام گذاری کرده و ارسال کنید. (برای مثال hw#_۱۲۳٤٥٦۷۸.pdf)
- به هیچ وجه تمرینی را از دیگران کپی نکنید. در صورت مشاهده تقلب و کپی در تمرینات، نمره هر دو طرف صفر در نظر گرفته می شود.

Written:

Q1- Determine the correction of the following statements by explaining the reason. (**12 point**)

a) Time complexity of deleting a node using the key of the node in a singly linked list, is of O(١).

b) Stack is the most suitable data structure if you only need to implement recursion in a programming language.

c) It is possible to implement one stack using only two queues.

d) The most appropriate data structure to print items of a queue in reversed order, is stack.

Q2- Suppose there are two singly linked lists both of which intersect at some point and become a single linked list. The head or start pointers of both the lists are known, but the intersecting node is not known. Also, the number of nodes in each of the lists before they intersect is unknown and may be different in each list. *List1* may have n nodes before it reaches the intersection point, and *List2* might have *m* nodes before it reaches the intersection point where *m* and n may be $m = n$,$m < n$ or $m > n$. Give an algorithm for finding the merging point. (with O(n + m))**(18 points)**

Q3- How to implement a stack which will support following operations in O(1) time complexity? Write your explanation and implement them.**(20 points)**

- Push which adds an element to the top of stack.
- Pop which removes an element from top of stack.
- Find Middle which will return middle element of the stack.
- Delete Middle which will delete the middle element.

Q4- For a given *K* value ($K > 0$) reverse blocks of *K* nodes in a list.**(12 points)**

Example: Input: 1 2 3 4 5 6 7 8 9 10. Output for different *K* values:

For $K = 2$: 2 1 4 3 6 5 8 7 10 9

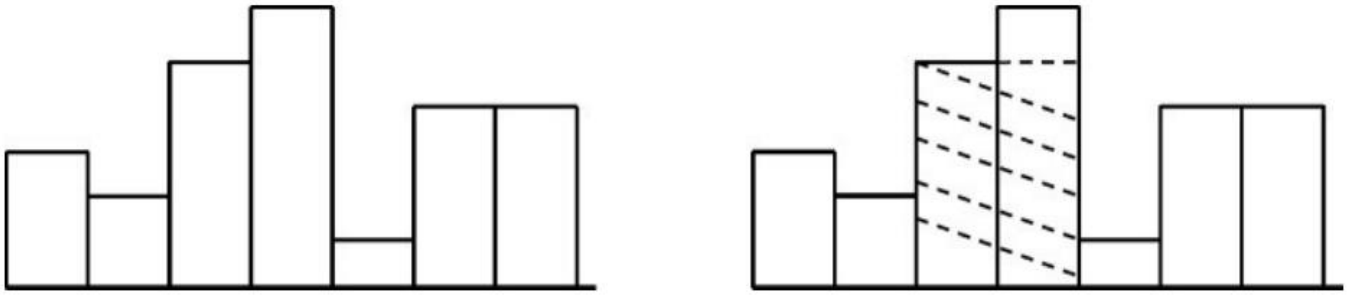
For $K = 3$: 3 2 1 6 5 4 9 8 7 10

For $K = 4$: 4 3 2 1 8 7 6 5 9 10

NOTE: You must solve this question with LinkedList.

Q5- A histogram is a polygon composed of a sequence of rectangles aligned at a common base line. For simplicity, assume that the rectangles have equal widths but may have different heights. For example, the figure on the left shows a histogram that consists of rectangles with the heights 3,2,5,6,1,4,4, measured in units where 1 is the width of the rectangles. Here our problem is: given an array with heights of rectangles (assuming width is 1), we need to find the largest rectangle possible. For the given example, the largest rectangle is the shared part.(**18 points**)

NOTE: You must solve this question using stack.



Implement:

Q1- Your friend is making a text editor for programmers. He is currently working on a feature that will find errors in the usage of different types of brackets. Code can contain any brackets from the set []{}(), where the opening brackets are [,{, and (and the closing brackets corresponding to them are],}, and).

For convenience, the text editor should not only inform the user that there is an error in the usage of brackets, but also point to the exact place in the code with the problematic bracket. First priority is to find the first unmatched closing bracket which either doesn't have an opening bracket before it, like] in](), or closes the wrong opening bracket, like } in ()[}. If there are no such mistakes, then it should find the first unmatched opening bracket without the corresponding closing bracket after it, like (in {}([]. If there are no mistakes, text editor should inform the user that the usage of brackets is correct.

Apart from the brackets, code can contain big and small Latin letters, digits and punctuation marks. More formally, all brackets in the code should be divided into pairs of matching brackets, such that in each pair the opening bracket goes before the closing bracket, and for any two pairs of brackets either one of them is nested inside another one as in (foo[bar]) or they are separate as in f(a , b)-g[c]. The bracket [corresponds to the bracket], { corresponds to }, and (corresponds to).**(20 points)**

Input Format: Input contains one string S which consists of big and small Latin letters, digits, punctuation marks and brackets from the set []{}().

Constraints: The length of S is at least 1 and at most 10^5.

Output Format: If the code in S uses brackets correctly, output "-1" (without the quotes). Otherwise, output the 1-based index of the first unmatched closing bracket, and if there are no unmatched closing brackets, output the 1-based index of the first unmatched opening bracket.

Sample 1.

Input:

```
[]
```

Output:

```
-1
```

Sample 2.

Input:

```
{}[]
```

Output:

```
-1
```

Sample 3.

Input:

```
{[}
```

Output:

```
3
```

Sample 4.

Input:

```
foo(bar);
```

Output:

```
-1
```

Sample 5.

Input:

```
foo(bar[i);
```

Output:

```
10
```