NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

II B. Tech I Semester Minor-1 Examinations, October 2021

CS201- Data Structures and Algorithms

Date: 04-October-2021 **Max. Marks: 30 Marks**

1. The following question uses a linked list consisting of nodes of the following type:

```
typedef struct Node{
        char data;
        struct Node *next;
} Node;
```

Variable *Node *head;* holds the header of the list.

Compute *mysteryFunction(str)* for *str = "<your name without spaces>", where for example if your name is "Gireesh Yellasiri" then str = "gireeshyellasiri".* Get an idea of what the mysteryFunction(str) does and *then* describe the output (return value) of *mysteryFunction(str)* in words for a given *str*.

[3 M]

Algorithm mysteryFunction(str)

1) Allocate space for the first node and set it to head.
2) Let a pointer p point to head and two pointers q and r be initialized to NULL.
3) Set int i=0 , n = strlen(str);
4) While i < n-1
    4.1) Set p→data = str[i].
    4.2) Allocate space for a new node and store the pointer to the node in p→next.
    4.3) Advance p to the next node and set the next field in the node pointed by p to null.
    4.4) Increment the value of i.
5) End while.
6) Set p→data = str[n-1].
7) Initialize p and q to point to head.
8) While (p)
    8.1) Advance p to the next node.
    8.2) If (p then
        8.2.1) Advance both p and q to the next node.
    8.3) End if
9) End while
10) return q→data;

End mysteryFunction

2. Given the following sequence of letters and asterisks: *EAS*Y*QUE***ST***IO*N***.* Consider the stack data structure, supporting two operations *push* and *pop*. For the above sequence each letter (such as E) corresponds to *push* operation inserting that letter into the stack and each asterisk (*) corresponds to *pop* operation on the stack. Construct a C code for implementing the above scenario and show the sequence of values returned by the *pop* operation.

[3 M]

3. Implement a queue data structure using a linked list and perform the following operations in $O(1)$ time. [3 M]
   (i) enqueue – enqueues an element at the head of the queue.
   (ii) dequeue – removes an element from the tail of the queue.

4. Assume that you have double linked list that contains duplicate elements. Construct an efficient C function combineDuplicates (list) that manipulates the double linked list as follows: the function modifies the list by merging any consecutive neighbour nodes that contains the same element value into a single node whose value is the sum of the merged neighbours. For instance, input and output are as follows:
   Input double linked list → {3, 3, 2, 4, 4, 4, -1, -1, 4, 12, 12, 12, 12, 48, -5, -5}
   Output → {6, 2, 12, -2, 4, 48, 48, -10} [3 M]
   Bonus points for those students who convert this output repeatedly, to merge duplicates in adjacent nodes until the adjacent elements in the list are different. For instance, {6, 2, 12, -2, 4, 48, 48, -10} is to be converted to {6, 2, 12, -2, 4, 96, -10}.

5. Consider the polynomial of the form:
   $$\mathrm{p(x)} = c_1 x^{e1} + c_2 x^{e2} + c_3 x^{e3} + \cdots$$
   Where $e1 > e2 > e3 > \cdots \geq 0$. such polynomials can be represented by a single linked list in which each node has three fields: one for coefficient, one for exponent, and one for pointing to the next node. Assume that you have two polynomials pointed to by *list1* and *list2*. Construct an efficient C function to multiply two polynomials and store the result in *list3*. [3 M]

6. Let A be an array of size n, containing positive or negative integers, with A[1] < A[2] < ... < A[n]. Design an efficient algorithm (should be more efficient than O(n)) to find an i such that A[i] = i provided such an i exists. What is the worst-case computational complexity of your algorithm? [3 M]

7. Construct an efficient algorithm to split a given circular single linked list pointed to by list, into two circular single linked lists pointed to by two pointers list1 and list2. These lists need to be approximately equal sized (in case of odd number of nodes in list) and equal-sized (in case of even number of nodes in list). This splitting must be done without counting the number of nodes in the linked list. [3 M]

8. Construct a time and space efficient C program to implement the stack data structure that performs the *push, and pop* operations using only a single queue data structure and using only its enqueue and dequeue operations. [3 M]

9. Construct an efficient C program to implement a circular queue data structure using arrays that performs the enqueue, dequeue, isFull, and isEmpty operations in O(1) time. [3 M]

10. You are given with the head pointer to the doubly linked list as the input. The linked list contains 'n' character elements with each character stored in one node. Construct an efficient C function to verify whether the string formed by the characters stored in the list, taken in the order from the head to the last node of the list, is a palindrome or not? [3 M]