National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

CS-2001: Data Structures

Serial No:

Sessional Exam-I

Part – A

Total Time: 35 Mins

Total Marks: 35

Monday, 25th September, 2023

Course Instructors

Dr. Usman Habib, Ms. Parisa Salma,
Ms. Kainat Iqbal, Mr. Atif Khurshid

_____
Signature of Invigilator

Student Name	Roll No.	Course Section	Student Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space, write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **five (5)** different printed pages including this title page. There are total of **two (2)** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Total
Marks Obtained			
Total Marks	**15**	**20**	**35**

Question 1

Part A	1	2	3	4	5
	operations	upper	next	First, Last	prefix
Part B	6	7	8	9	10
	C	B	B	A	B

Part A. Complete the blanks. Write your answers in the table above. **[5 marks]**

1. An Abstract Data Type is a collection of data and associated ____.
2. Big-O notation is used to describe the ____ bound of an algorithm's complexity.
3. In a singly linked list, each node contains a data element and a pointer to the ____ node in the list.
4. A stack is a ____ In ____ Out data structure.
5. In ____ notation, the operators are written before the operands.

Part B. Choose the correct option and write your answers in the table above. **[5 marks]**

6. Consider the following array. mention the position of an element at which binary search has the best case scenario? int arr[6] = {1,2,3,4,5,6}

   A. The first element, arr[0] (position 0).
   B. The last element, arr[5] (position 5).
   C. The middle element, arr[2] (position 2).
   D. The element arr[3] (position 3).

7. Consider an implementation of unsorted singly linked list. Suppose you access the linked list only with head pointer. Given the representation, which of the following operation can be implemented in O(1) time?

   i. Insertion at first
   ii. Insertion at last
   iii. Deletion at first
   iv. Deletion at last

   A. Option i and ii
   B. Option i and iii
   C. Option i, ii and iii
   D. Option i, iii and iv

8. What will be the output of the following code snippet for the given list 1->2->3->4->5->6?

```
void solve(struct node* start)
{
    if (start == NULL)
        return;
    cout<< start->data;
    if (start->next != NULL)
        solve(start->next->next);
    cout<< start->data;
}
```

A. 1 2 3 4 5 6
B. 1 3 5 5 3 1
C. 1 3 5 1 3 5
D. 2 4 6 1 3 5

9. In the given code, the value is inserted at location. What if the given location does not exist in the linked list?

```
void insertAtLocation(int loc, int val) {

    Node* temp = new Node;
    temp->data = val;
    temp->next = NULL;
    Node* pre = head;
    Node* cur = head->next;
    for (int i = 1; i < loc - 1; i++) {
        pre = cur;
        cur = cur->next;
    }
    pre->next = temp;
    temp->next = cur;
}
```

A. It will throw exception
B. It will run smoothly.

10. In the question above (4), is there any need of exception handling? If yes, what will be the condition?
    A. Not required.
    B. `if (cur->next == NULL) { break; }`
    C. `if (cur == NULL) { break; }`

Part C. Write the worst-case time complexity, in terms of Big-O, of the following pieces of code?
[5 marks]

No.	Code	Complexity

1	`void func1(int* arr, int n)`
	`{`
	`  for(int i = 0; i < n; i++)`
	`  {`
	`  for(int k = i; k < n; k++)`
	`  {`
	`  cout<< arr[i];`
	`  }`
	`  }`
	`}`
		$O(n^2)$

2	`void func1(int* arr, int n)`
	`{`
	`  for(int i = 0; i < n; i++)`
	`  {`
	`  for(int k = 0; k <n; ++k)`
	`  {`
	`  cout<<arr[i];`
	`  }`
	`  }`
	`}`
	
	`void func2(int* arr, int n)`
	`{`
	`  for(int i = n-1 ; i >= 0; i = i-3)`
	`  {`
	`  cout<<arr[i]);`
	`  }`
	`}`
	
	`void func3(int* arr, int n)`
	`{`
	`  for(int i = 0; i < n; i++)`
	`  {`
	`  func1(arr,n);`
	`  func2(arr,n);`
	`  }`
	`}`
		Calculate time complexity of func3 only.
		
		$O(n^3)$

Question 2

Part A: Recall algorithm to convert infix expression (with parenthesis) to postfix. Given the following infix expression, show the complete working of the algorithm in the table below. For each token/symbol, write down the postfix string and opstk. **[10 marks]**

Expression: ((5+(2+3*2/5)*2)*2+5)

Symbol	Postfix String	Operator Stack
(		(
(		((
5	5	((
+	5	((+
(	5,2	((+(
2	5,2	((+(
+	5,2	((+(+
3	5,2,3	((+(+
*	5,2,3	((+(+*
2	5,2,3,2	((+(+*
/	5,2,3,2*	((+(+/
5	5,2,3,2*5	((+(+/
)	5232*5/+	((+
*	5232*5/+	((+*
2	5232*5/+2	((+*
)	5232*5/+2*+	(
*	5232*5/+2*+	(*
2	5232*5/+2*+2	(*
+	5232*5/+2*+2*	(+
5	5232*5/+2*+2*5	(+
)	5232*5/+2*+2*5+	

Part B: Evaluate the given postfix expression using the algorithm, you study in the class. **[10 marks]**

Expression: 5 4 + 7 2 - * 3 2 * 3 / +

Symbol	Operand 1	Operand 2	Value	Operand Stack
5				5
4				5,4
+	5	4	9	9
7				9,7
2				9,7,2
-	7	2	5	9,5
*	9	5	45	45
3				45,3
2				45,3,2
*	3	2	6	45,6
3				45,6,3
/	6	3	2	45,2
+	45	2	47	47

National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

CS-2001: Data Structures

Serial No:

Sessional Exam-I

Part – B

Total Time: 25 Mins

Total Marks: 25

Monday, 25th September, 2023

Course Instructors

Dr. Usman Habib, Ms. Parisa Salma,
Ms. Kainat Iqbal, Mr. Atif Khurshid

_____
Signature of Invigilator

_____	_____	_____	_____
Student Name	Roll No.	Course Section	Student Signature

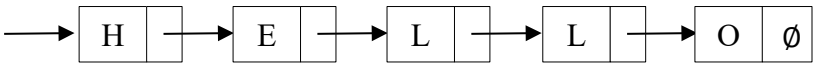
DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space, write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **four (4)** different printed pages including this title page. There are total of **two (2)** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Total
Marks Obtained			
Total Marks	**10**	**15**	**25**

In C programming, a string is an array of characters terminated with the null character '\0'. Similarly, we can also represent strings as linked lists of characters.

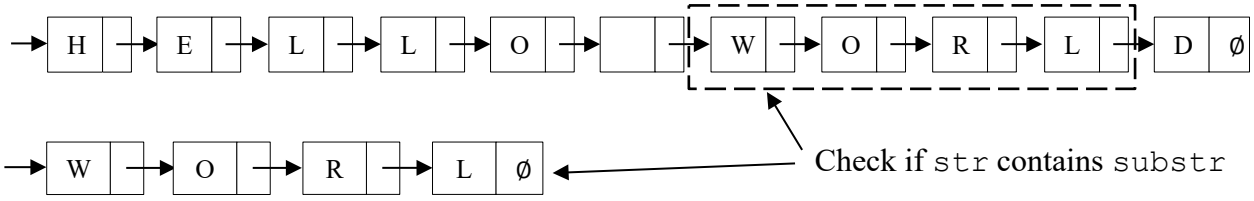


Consider the following definition of a LinkedString structure where strings are implemented using a singly linked list of character nodes CharNode.

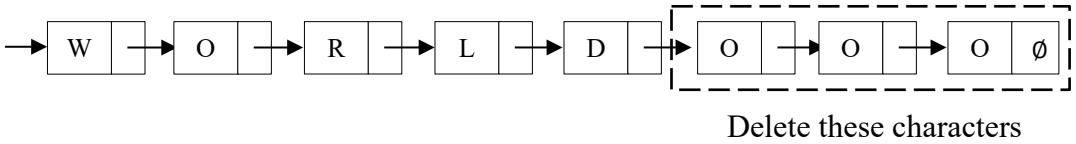
```
struct CharNode {
     char      data;
     CharNode* next;
};

struct LinkedString {
     CharNode* head;
     int       length;
};
```

Implement the following string methods for LinkedStrings.

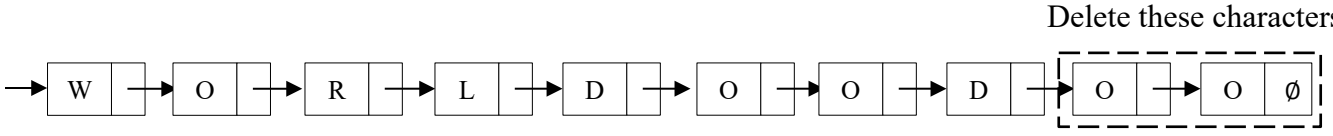
A. isSubstring: This function checks if a string A contains another string B as a subset. For example, "Hello World" contains "Worl" as a substring.



B. rightStrip: This function removes specified trailing characters from the end of a string. For example, removing the trailing character 'o' from "Worldooo" results in the string "World".



Notice that only the rightmost occurrences of the character 'o' are removed. So applying this function on "Worldoodoo" will return 'Worldood'.



However, applying the function on 'World' will do nothing because 'o' does not occur at the end of this string.

Question 1

The function should take two LinkedStrings, `str` and `substr`, as arguments and check if the entire sequence of characters in `substr` occurs anywhere in `str`, in which case it should return true. Otherwise, it should return false. **[10 marks]**

```
bool isSubstring(LinkedString str, LinkedString substr) {

    CharNode* strStart = str->head;

    while (strStart != NULL) {

        CharNode* strCurrent = strStart;
        CharNode* substrCurrent = substr->head;

        while (strCurrent != NULL && substrCurrent != NULL &&
            strCurrent->data == substrCurrent->data) {

            strCurrent = strCurrent->next;
            substrCurrent = substrCurrent->next;

        }
        if (substrCurrent == NULL)
            return true;

        strStart = strStart->next;
    }

    return false;



}
```

Question 2

The function should take a LinkedString `str` and a character `c` as arguments and remove all occurrences of the character `c` from the end of the string `str`. **[10+5 marks]**

Note: You will get the 5 marks if your algorithm requires a single pass through the linked list.

```
void rightStrip(LinkedString &str, char c){

      CharNode* current = str->head;
      CharNode* lastNonMatching = NULL;

      while (current != NULL) {

            if (current->data != c) {
                  lastNonMatching = current;
            }
            current = current->next;
      }

      if (lastNonMatching != NULL) {

            CharNode* firstMatching = lastNonMatching->next;
            lastNonMatching->next = NULL;

            while(firstMatching != NULL) {

                  CharNode* ptrForDeletion = firstMatching;
                  firstMatching = firstMatching->next;
                  delete ptrForDeletion;
                  str->length--;

            }
      }
}
```