Air University

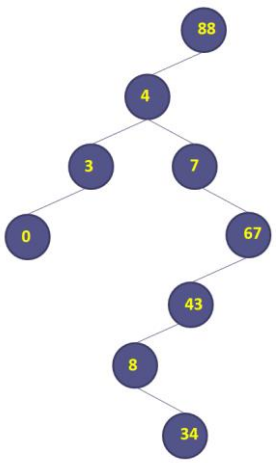
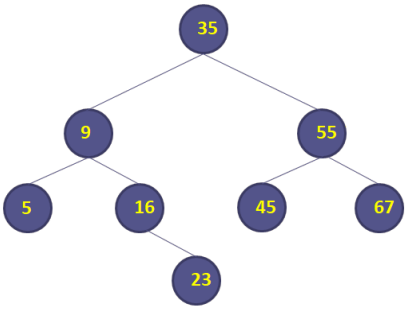
(Mid-Term Examination: Fall 2022)

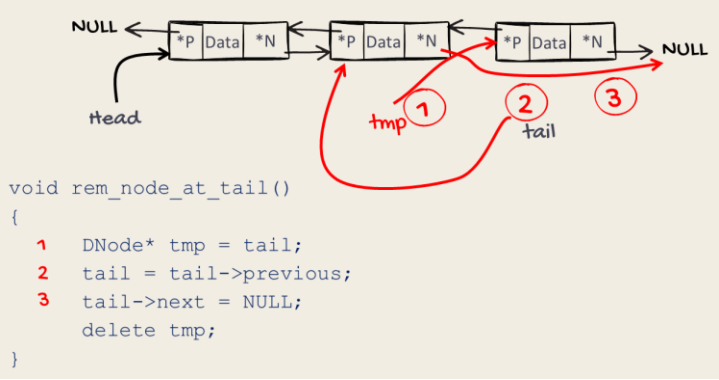
Subject:	**Data Structures and Algorithms**	Total Marks:	**100**
Course Code:	**CS-214**	Date:	
Class:	**BS-CYS**	Time:	
Semester:	**III**	Duration:	**2 Hours**
Section:	**A**	FM Name:	**Dr. Mohammad Imran**

HoD Signatures:_____ FM Signatures:_____

Note:

- **All questions must be attempted.**
- **This examination carries 25% weight towards the final grade.**
- **Return the question paper with the answer sheet**

	Q. No. 1 (CLO 1)	**25 Marks**
a	We perform the following operations on an empty stack: `push (9)`, `push (3)`, `pop`, `push (7)`, `push (2)`, `pop`, `pop`, `pop`, `push (6)`, `pop`. Write the sequence of popped out values. **Answer:** 3, 2, 7, 9, 6	5
b	Generate Binary Search Trees for the following sequence of numbers: (i) 88, 4, 7, 3, 67, 43, 8, 0, 34 (ii) 35, 9, 5, 16, 23, 55, 45, 67 **Answer:** (i)  (ii)  	5+5=10
c	In order to delete the last node in a linked list, we have to traverse the whole list to get to the last node and it requires *O*(n) operations. How can we modify the node class or the linked list class so that we can delete the last node in constant time, i.e., *O*(1)? Draw a diagram to explain your answer. **Answer:**	10

	Basically we just convert the linked list into a doubly linked list! We need to do two things: We add a pointer to previous node in the node class, which makes it a doubly linked list node, and then we add a pointer to last node (the *tail* pointer), to the doubly linked list class. As we discussed in the class, deleting the tail node of a doubly linked list is an $O(1)$ operation!
	 
	
	`void rem_node_at_tail()`
	`{`
	`1 DNode* tmp = tail;`
	`2 tail = tail->previous;`
	`3 tail->next = NULL;`
	`delete tmp;`
	`}`	

	Q. No. 2 (CLO 2)	**25 Marks**
a	For each of the following scenarios choose the "best" data structure from the following list: an Array, Linked List, Doubly Linked list, Circular Linked List, Stack, Queue, or Tree. In each case, justify your answer briefly. a) A restaurant needs to first serve the customers who come first **Answer:** Queue b) A list must be maintained so that any element can be accessed randomly **Answer:** Array c) A program needs to mainatin a history of the operations it performed, so that it can undo or redo any operation **Answer:** Stack or Doubly linked list. d) The size of data to be stored is unknown. The entries need to be entered as they come in. Entries must be deleted when they are no longer needed. It is important that structure has flexible memory management. **Answer:** Linked list e) A person wants to record names of his ancestors all the way to Adam **Answer:** Tree	10
b	Assume a function f1 is executing and it calls another function f2. f1 needs to store its local variables in memory so that these can be loaded back when f2 completes. Suppose f2 calls another function f3. Now the local variables for f2 also need to be stored in memory so that it can resume its operation when f3 completes. So variables for f1 will be stored first, then variables for f2, and then variables for f3. But the variables for f3 will be retrieved first, then variables for f2, and then variables for f1. (i) What data structure is suitable for storing local variables in memory for such a scenario? **Answer:** Since the variables need to be retrieved in the opposite order in which these were stored, therefore this scenario requires a Last In First Out data structure – a stack. (ii) What operations will be required in such a data structure? **Answer:** We need to have a function for storing the local variables of a function at the top	3+4+8=15

	of the stack – the push operation, and then another function for retrieving the variables which are currently on the top of the stack – the pop operation. (iii) How can such a data structure be implemented using a linked list? (Don't write the code, just assume that you already have a linked list and explain how you would use the functions of linked list in the given scenario) **Answer:** If we already have a linked list class and it has the functions to add a node at the head and delete the node at head, then we can use this list for implementing our stack. The function for adding a node at the head will be used in push operation, and the function for deleting the node at head will be used in pop operation. But since the pop operation should return the value of the node being deleted, therefore we need to first save the value of the head node in some variable before deleting the head node.	
	Q. No. 3 (CLO 3)	**25 Marks**

a	Consider the following declarations of an array and a pointer, and answer the questions:
	`int iArray[10] = {10,20,30,40,50,60,70,80,90,100};`
	`int *iPtr = iArray;`
	(i) What will the following line of code do?
	`cout << *(iPtr-1);`
	(ii) What will be output of the following code?
	`cout << *(iPtr+5) << ", ";`
	`cout << (*iPtr)+1 << ", ";`
	`iPtr += 3;`
	`cout << *(++iPtr) << ", ";`
	`cout << *(--iPtr);`
	Answer:
	(i) We are trying to read data outside the memory allocated for the array! This could print some garbage value, or even crash the program.
	(ii) 60, 11, 50, 40
		2+8=10

b	Write the algorithm/ pseudocode for the **`find(int x)`** method for a linked list that returns `true` if there is a node in the list with value **x**, and returns `false` otherwise.
	Answer:
	`bool find(int x)`
	`{`
	`  Node *tmp = head;`
	`  while (tmp != NULL)`
	`  {`
	`  if (tmp->data == x)`
	`  return true;`
	`  else`
	`  tmp = tmp->next;`
	`  }`
	`  return false;`
	`}`
		15

	Q. No. 4 (CLO 4)	25 Marks

a	Calculate the time complexity of following code. Show your working for `if-else` and `for` loops. **Do NOT write the whole code on the answer sheet**! Just show your working here and provide the answer on the answer sheet.
	
	`int a=0, b=0;`
	`for (i=0; i<n; i++) {`
	`  if (i<k) {`
	`  a = i;`
	`  b = n – i;`
	`  }`
	`  else {`
	`  for (j=k; j<n; j++) {`
	`  a = a + j;`
	`  b = a – k;`
	`  }`
	`  }`
	`}`
	
	Answer:
	Number of operations in *if* statement: 4
	Complexity of *if* statement: $O(4) = O(1)$
	Number of operations in *else* statement: 2 (*for j* loop testing)+2(first statement) + 2(second statement) = 6. But *for j* loop runs *n* times in the worst case because we don't know the value of *k*
	Complexity of *else* statement: $O(6n) = O(n)$
	Complexity of combined *if-else* statement: Max ($O(1)$, $O(n)$) = $O(n)$
	for i loop executes *n* times, and complexity of each loop is $O(n)$
	Therefore complexity of *for i* loop = $O(n) * O(n) = O(n^2)$
	Declarations are not considered, so the complexity of the given code is $O(n^2)$
		10

b	(i) Write down a recursive algorithm for finding the maximum number in an array.
	(ii) Calculate the time complexity of your algorithm
	
	Answer:
	`Algorithm max_array (int array[], int n)`
	
	`  if n == 1`
	`  Return array[1]`
	`  Else`
	`  Return max (array[n], max_array(array, n-1))`
	
	The time complexity of this algorithm is $O(n)$.
		10+5=15

********** End of Question Paper **********