VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Fall Semester 2022-2023 Continuous Assessment Test – I

SLOT: F1+TF1

Programme Name &Branch :BCB, BCE, BCI, BCT, BDS, BKT

Course Name & code: Data Structures and Algorithms - BCSE202L

Class Number (s):3295

Faculty Name (s): Prof. Saravanakumar K, Prof. TusarKanti Mishra + 16

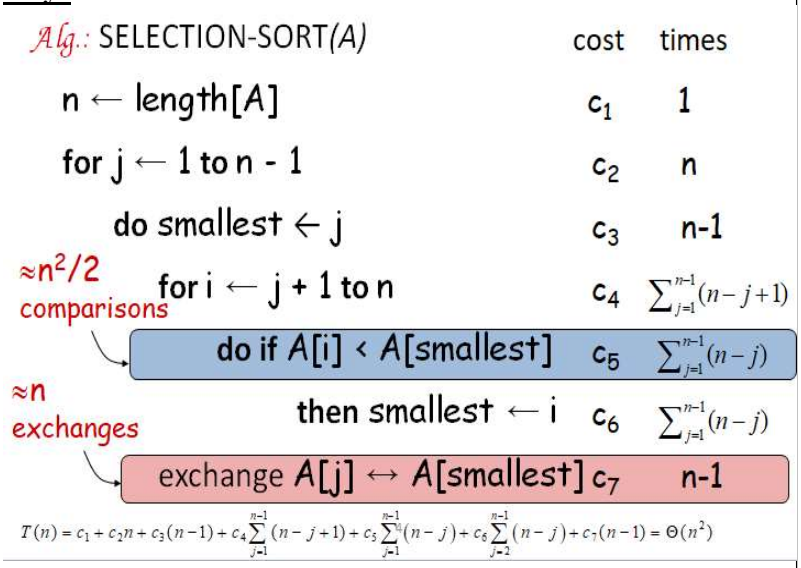
Exam Duration: 90 Min. **Maximum Marks: 50**

General instruction(s):

Specify if any printed material may be permitted

Any other specific instruction

Q.No.	Question	Max Marks	CO	BL
1.	Answer the following: (i) What do you mean by a good algorithm? **Key:** Explanation on Input, Output, Finiteness, Definiteness, Effectiveness. (ii) Why it is necessary to analyze algorithm rather than the program? **Key:** Algorithm analysis is independent of software and hardware resources, Algorithm analysis provides genuine comparison among distinct solutions to common problems, Algorithm analysis is independent of the experimental timeline, Program analysis does not adhere to any of these major points. (iii) Discuss the role of asymptotic notations with an example. **Key:** Determination of order of magnitude of a statement. Asymptotic notations allow us to analyze an algorithm's running time by identifying its behavior as the input size for the algorithm increases. This is also known as an algorithm's growth rate. Does the algorithm suddenly become incredibly slow when the input size grows? Does it mostly maintain its quick run time as the input size increases? Asymptotic Notation gives us the ability to answer these questions. To get the most from any program, we need to follow some structure and need to find out the notation of the problem.	10 [3 + 3 + 4]	CO 1	BL 1

	As examples, student should write about big oh, omega, and theta notations.			
2.	(i) Design the pseudocode having time complexity not more than $O(Log_2N)$ for searching an ITEM in a given array. Verify that it meets the criteria for the said time complexity. **Key:** Binary search algorithm to be written. This should be followed by time complexity analysis. (ii) Write down the algorithm for selection sort and analyze the same for time complexity. **Key:** 	10 [5 + 5]	CO 3	BL 2
3.	Write the steps or algorithm to convert infix expression into postfix expression. Using stack, convert(A – B) / ((D + E) * F) – G into postfix notation. Show the conversion process in Table 1. **Input Character/Token** | **Operation (PUSH/POP)** | **Contents of Stack** | **Result/Output** Table 1 **Key:** **Algorithm** Step 1 : Scan the Infix Expression from left to right. Step 2 : If the scanned character is an operand, append it with final Infix to Postfix string. Step 3 : Else, Step 3.1 : If the precedence order of the scanned(incoming) operator is greater than the precedence order of the operator in the stack (or the stack is empty or the stack contains a '(' or '[' or '{'), push it on stack. Step 3.2 : Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the	10	CO 2	BL 3

<table>
<tr><td></td><td>scanned operator in the stack.)
Step 4 : If the scanned character is an ‘(‘ or ‘[‘ or ‘{‘, push it to the stack.
Step 5 : If the scanned character is an ‘)’or ‘]’ or ‘}’, pop the stack and and output it until a ‘(‘ or ‘[‘ or ‘{‘ respectively is encountered, and discard both the parenthesis.
Step 6 : Repeat steps 2-6 until infix expression is scanned.
Step 7 : Print the output
Step 8 : Pop and output from the stack until it is not empty.

Equivalent Postfix expression of the given expression
AB-DE+F*/G-</td><td></td><td></td><td></td></tr>
<tr><td>4.</td><td>Assume that the operation ENQUEUE (CQ, x) inserts an item x into a circular queue CQ and another operation DEQUEUE (CQ) deletes an item from CQ in FIFO manner. Draw the circular queue of size 6. Illustrate the working for the following eight commands (in the given order) over that circular queue (illustrate with diagram for execution of every command), also give the values of the variables FRONT and REAR on execution of each command;
(i) ENQUEUE (CQ, 10); (ii) ENQUEUE (CQ, 20); (iii) ENQUEUE (CQ, 30); (iv) DEQUEUE (CQ);
(v) ENQUEUE (CQ, 40); (vi) ENQUEUE (CQ, 50); (vii) ENQUEUE (CQ, 60); (viii) DEQUEUE (CQ);

Answers are expected as per the format given in Table 2;
<u>Key:</u>
Front = Rear = -1 when queue is empty
Front = (front+1)%6; (while ENQUEUEing for the first time or while DEQUEUEing)
Rear = (rear+1)%6; (while ENQUEUEing)
<table><tr><th>Command</th><th>Circular queue</th><th>Front</th><th>Rear</th></tr><tr><td>ENQUEUE(CQ,10)</td><td>Expected to draw the appropriate circular queue here</td><td>0</td><td>0</td></tr><tr><td>ENQUEUE(CQ,20)</td><td></td><td>0</td><td>1</td></tr><tr><td>ENQUEUE(CQ,30)</td><td></td><td>0</td><td>2</td></tr><tr><td>DEQUEUE(CQ)</td><td></td><td>1</td><td>2</td></tr><tr><td>ENQUEUE(CQ,40)</td><td></td><td>1</td><td>3</td></tr><tr><td>ENQUEUE(CQ,50)</td><td></td><td>1</td><td>4</td></tr><tr><td>ENQUEUE(CQ,60)</td><td></td><td>1</td><td>5</td></tr><tr><td>DEQUEUE(CQ)</td><td></td><td>2</td><td>5</td></tr></table>Table 2</td><td>10</td><td>CO 2</td><td>BL 3</td></tr>
<tr><td>5.</td><td>Explain in detail about the Quick Sort algorithm followed by suitable example and running time description.</td><td>10</td><td>CO 3</td><td>BL 2</td></tr>
</table>

Key:

Quicksort

//start –> Starting index, end --> Ending index

Quicksort(array, start, end)

```
{
if (start < end)
  {
pIndex = Partition(A, start, end)
Quicksort(A,start,pIndex-1)
Quicksort(A,pIndex+1, end)
  }
}
```

Partition function

```
partition (array, start, end)
{
   // Setting rightmost Index as pivot
pivot = arr[end];

   i = (start - 1)  // Index of smaller element and indicates the
                    // right position of pivot found so farfor (j = start; j <= end- 1; j++)
   {
      // If current element is smaller than the pivot
if (arr[j] < pivot)
      {
i++;    // increment index of smaller element
swaparr[i] and arr[j]
      }
   }
swaparr[i + 1] and arr[end])
return (i + 1)
}
```

Any example can be written and discussed here.

Worse case time complexity

$N+(N-1)+(N-2)+(N-3)+...+2 = [(N(N+1)/2)-1] = O(N^2)$

Worst case may happen in one of the following cases;

When the input array is already sorted

When the given array is reverse sorted

When all the elements in the given array are the same