Roll Number:______________________ Number of Pages: 02

Thapar Institute of Engineering and Technology, Patiala

Department of Computer Science and Engineering

MID SEMESTER EXAMINATION

B. E. (Second Year): Semester-II (2018/19)	Course Code: UCS406 Course Name: Data Structures and Algorithms
March 15, 2019	Friday, 10:30 Hrs – 12:30 Hrs
Time: 2 Hours, M. Marks: 25	Name of Faculty: SMG, SUG, SP, TBH, RKR, RAH, ASG, ANK

Note: Attempt all questions (sub-parts) in sequence. Assume missing data, if any, suitably.

Q1. Perform the following operations using stacks. Show contents of the stack at each intermediate step.

(a) Convert the given infix expression into an equivalent postfix expression. **(2)**

A – B – C * (D + E / F – G) – H

(b) Compute the value of the postfix expression obtained in Q1.(a) for **(2)**

A = 45, **B = F = 2**, **C = 5**, **D = 8**, **E = 6**, **G = 4**, and **H = 3.**

Q2. Write a complete algorithm/pseudo-code to implement any one of the following: **(3)**

Quicksort sorting algorithm **OR** Mergesort sorting algorithm

Q3. (a) Solve the following recurrence relation. **(1)**

$$T(n) = \begin{cases} 0 & , n = 0 \\ T(n-1) + 2n - 1 & , n > 0 \end{cases}$$

(b) Find the recurrence relation and solve it for the function given in **Fig. 1**. **(2)**

```
int power(int x, int n)
{ if (n==0)
    return 1;
  else if (n==1)
    return x;
  else if ((n%2)==0)
    return power(x, n/2)*power(x, n/2);
  else
    return power(x, n/2)*power(x, n/2);
}
```

Fig. 1

```
1.  for (int k = 1; k <= 7; k++)
2.    Q.enqueue(k);
3.  for (int k = 1; k <= 4; k++)
4.  {
5.    Q.enqueue(Q.dequeue());
6.    Q.dequeue();
7.  }
```

Fig. 2

Q4. (a) Let $f(n) = 7n + 8$ and $g(n) = n$. Is $f(n) = O(g(n))$? **(1)**

If yes, then determine the values of n_0 and c showing all intermediate steps.

If no, then justify your answer with appropriate explanation.

(b) An algorithm **ALGO** consists of two tuneable sub-algorithms **ALGO$_A$** and **ALGO$_B$**, which have to be executed serially. Given any function $f(n)$, one can tune **ALGO$_A$** and **ALGO$_B$** such that one run of **ALGO$_A$** takes time $O(f(n))$ and **ALGO$_B$** takes time $O(n/f(n))$. For the given scenario, determine the smallest growing function $f(n)$ which minimizes the overall runtime of **ALGO**. **(2)**

Q5. Let **Q** be a circular array-based queue capable of holding **7** numbers. Execute the code snippet given in **Fig. 2**. After each execution of the **for loop in lines 3 to 7**, give the values of *front* pointer, *rear* pointer, and *valid contents* of **Q**, i.e. elements in between the *front* and the *rear* pointers. **(2)**

Q6. Let **S** be an empty stack and **Q** be a queue having *n* numbers. **isEmpty(Q)** or **isEmpty(S)** returns **true** if **Q** or **S** is empty, else returns **false**. **top(S)** returns the number at the *top* of **S** without removing it from **S**. Similarly, **front(Q)** returns the number at the *front* of the queue **Q** without removing it from **Q**. **(2)**

Determine the best- as well as the worst-case running time of an algorithm shown in **Fig. 3**. Justify your answers giving suitable examples. [*Hint: Use n <= 4*].

```
while (!isEmpty(Q))
{ if (isEmpty(S) || top(S) >= front(Q))
  { S = push(S,front(Q));
    Q = dequeue(Q);
  }
  else
  { Q = enqueue(Q,top(S));
    S = pop(S);
  }
}
```

Fig. 3

```
/*  Integer  n  is  the  number  of
elements in an array A[0..n-1]. */

void module(int *A, int n, int k)
{ int temp, i, j;
  for (j = 0; j < k; j++)
  { temp = A[n-1];
    for (i = n - 1; i > 0; i--)
        A[i] = A[i - 1];
    A[i] = temp;
  }
}
```

Fig. 4

Q7. Answer the following questions with respect to the function given in **Fig. 4**. **(2)**

(a) What is the purpose of designing it? [*Hint: Use n <= 5, 1 <= k <= n*]

(b) What is its complexity?

(c) Is answer to Q7.(b) dependent on the value of **k**? If yes, then for k > n suggest a single line modification in the given function to maintain the identified time complexity as in Q7.(b). If no, then give suitable justification with examples for the identified independency.

Q8. Given a singly linked list (**LL1**) having **2*n** nodes (*n* ≥ 1). **(6)**

(a) Write an algorithm/pseudo-code to create two linked lists (**LL2** and **LL3**) each having *n* – 1 nodes. **LL2** and **LL3** are respectively formed by adding values of consecutive odd-positioned and even-positioned nodes in **LL1**.
Note: Position of first node in LL1 is one.
Example: n = 3, LL1: 1 → 2 → 3 → 4 → 5 → 6
LL2: 4 → 8
LL3: 6 → 10

(b) Write an algorithm/pseudo-code to combine **LL1** with **LL2** and **LL3** (formed in Q8.(a)). Nodes of **LL2** and **LL3** are to be placed at alternative positions in first-half and last-half of **LL1**. Create a new node **MID** that contains sum of first and last node values of **LL1** and place it in the middle of the **updated LL1** as shown in **Fig. 5**.
Note: Creation of new node is not allowed, only reposition the existing nodes.
Example: In continuation with example of Q8.(a)
MID: 7
Updated LL1: 1 → 4 → 2 → 8 → 3 → 7 → 4 → 6 → 5 → 10 → 6
LL2: NIL and LL3: NIL

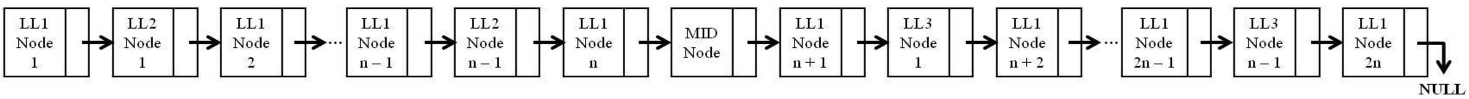

Fig. 5

---**ALL THE BEST**---