INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR
Department of Computer Science and Engineering

Algorithms-I (CS21003)
Class Test-2 (Spring)

Students: 92 *Time:* 10:15-11:30 am (1 hour 15 mins.) *Date:* Tue, March 14, 2023
Place: NR-122 *Marks:* 30

1. Answer all the questions on the question paper itself in the spaces provided.
2. For **Questions 1—8**, you have to write only the answers. No explanation is needed. No part marks; you get full marks only if your answer is fully correct.
3. Unless mentioned otherwise, *time complexity of a problem* P means the worst-case time complexity by the best possible algorithm for P.
4. r denotes the last digit of your roll number, and $r \bmod p$ is the remainder obtained when r is divided by a positive integer p. In every question where r appears, you have to write the numerical answer after substituting the value of r, failing which no marks will be awarded.

Roll no: __________ **Name:** ____________________ **Section:** _____

1. Which ones of the following is/are correct about a binary heap? (put tick marks) 2

   (A) Heap is realised as an array
   (B) Heap is a complete binary tree
   (C) Heap is a priority queue
   (D) Building a heap of n elements takes $\Theta(n \log n)$ time
   (E) Inserting a new element in a heap takes logarithmic time

2. There are k sorted lists, each containing n elements. How quickly can you construct a sorted list containing the k smallest elements of all these lists? You may assume that all kn elements are distinct. (put a tick mark) 2

   (A) $O(kn)$ time
   (B) $O(kn \log k)$ time
   (C) $O(k \log k)$ time
   (D) $O(k^2 n)$ time
   (E) $O(kn^2)$ time

3. There are k sorted lists, each containing n elements. How quickly can you get a sorted list containing all these kn elements? You may assume that all kn elements are distinct. (put a tick mark) 2

   (A) $O(kn)$ time
   (B) $O(k^2 n)$ time
   (C) $O(kn \log(kn))$ time
   (D) $O(kn \log k)$ time
   (E) $O(kn \log n)$ time

4. At most how many nodes have height h in any n-element binary heap? (The bottom-most level has height 0 and the topmost level has maximum height.) 2

   $\lceil n/2^{h+1} \rceil$

5. The *transpose* of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$ where the edges of G are reversed, i.e., $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$. Given G in adjacency-list representation, how quickly can you find whether G^T has a cycle? 2

   $O(V + E)$ time

6. $G = (V, E)$ is a weighted, undirected graph with $|V| = 10 + r \bmod 5$, and the vertices labeled by the numbers $1, 2, \ldots, |V|$. Between every two vertices i and j, there is an edge with weight $i + j - 2$. What will be the total weight of its MST? (You have to write its integer value.) 3

   $1 + 2 + \cdots + (n - 1) = n(n - 1)/2$, where $n = 10 + r \bmod 5$.

7. Prove or disprove: "Given any undirected graph G, a shortest path between two nodes of G is necessarily a part of some MST of G." 3

   False. Counterexample: G is a 4-vertex cycle with edge weights $1, 2, 3, 5$.

8. Let $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$, and $c(x) = a(x)b(x)$. Given a_i's and b_i's, how many scalar multiplications are enough to compute the seven coefficients of $c(x)$? 4

   9.

   For brevity, represent the product by tuples of exponents of x:

   $$(3, 2, 1, 0) * (0, 1, 2, 3) = ((3, 2), (1, 0)) * ((3, 2), (1, 0)$$

   Here, '$*$' denotes the multiplication between tuples of two or more elements. Now, find the scalar multiplications as follows.

   Step 1: $(3, 2) * (3, 2)$ → 3 mul exp = **6**, **5**, 4

   Step 2: $(1, 0) * (1, 0)$ → 3 mul exp = 2, **1**, **0**

   Step 3: $(3, 2) * (1, 0)$, $(1, 0) * (3, 2)$ → 3 mul exp = 4, **3**, 2

   bold font means completed in one step

   Step 3 details: There are $4+4 = 8$ scalar-multiplication terms in total, which can be divided into 3 groups for the exponents $4, 3, 2$, and can be computed in the order $4, 2, 3$, as follows:

   a) exp $= 4$: $3 \cdot 1 + 1 \cdot 3$ comes from $(3, 1) * (1, 3)$ using 1 mul, as $3 \cdot 3$ and $1 \cdot 1$ are already available (Steps 1 and 2).

   b) exp $= 2$: $2 \cdot 0 + 0 \cdot 2$ comes from $(2, 0) * (0, 2)$ using 1 mul, as $2 \cdot 2$ and $0 \cdot 0$ are already available (Steps 1 and 2).

   c) exp $= 3$: $3 \cdot 0 + 2 \cdot 1 + 1 \cdot 2 + 0 \cdot 3$ comes from $(3, 2, 1, 0) * (0, 1, 2, 3)$ using 1 mul, as the subtrahends are already available (Steps 1, 2, 3a, 3b).

9. You are given an array **A** of **n** integers (both positive and negative). You are required to identify the window in **A** starting at **l** and ending at **r** so that the sum **A[l]+...+A[r]** is maximum. This information is stored in records (structures) of type **rsTyp** defined through the **typedef** below. You need to fill up the blanks in the code below to achieve this functionality. Read the comment following the function declaration to know the functionality of each defined function.

```
typedef struct rsTag { // return type for sum and range
    int s; /* sum */ int l, r; // left and right indices of range
} rsTyp;

rsTyp maxSumRngLR (int n, int A[n], int d) { // n>=1, d = 1 or d = -1
// find maxSum window starting at 0 if d=1 or at n-1 if d=-1
// maxSumRngLR (8, A={-7, 3, -1, 4, -2, 8, -2, -5}, 1) -> (5, 0, 5)
// maxSumRngLR (8, A={-7, 3, -1, 4, -2, 8, -2, -5},-1) -> (5, 1, 7)
    rsTyp rv; // return value
    int i,iX,s,sX; if (d==1) { rv.l=iX=0; } else { rv.r=iX=n-1; } s=sX=A[iX];
    for (i=(d==1?1:n-2) ; (d==1?i<n:i>=0) ; i+=d) {

        s += A[i];
        if (s > sX) {
            sX=s;
            iX=i;
        }                                                                 2
    }
    if (d==1) { rv.r = iX; } else { rv.l = iX; } rv.s = sX; return rv;
}

rsTyp maxSumRngMM (int n, int A[n], int m) { // n>=1
// Routine to find the max sum window spanning the midpoint m
// utilising maxSumRngLR () so that l <=m <= r, where l and r are
// the start and end indices of the window
    rsTyp rv; // return value
    rsTyp rvL = maxSumRngLR ( /* left part ending at the middle */
        ______n-m______, ______A+m______, ______-1______);                1
    rsTyp rvR = maxSumRngLR ( /* right part starting at the middle */
        ______m______, ______A______, ______1______);                     1
    if (rvL.s > 0 && rvR.s > 0) { /* now the overall window */
        rv.s = ______rvL.s + rvR.s______;
        rv.l = ______rvL.l______; rv.r = ______rvR.r + m______;           1
    } else {
        // other cases ... variations of above ... no need to write
    }
    return rv;
}
```

```
rsTyp maxSumRngDC (int n, int A[n]) { // n>=1
// This the divide and conquer scheme to find the maximum sum window
    rsTyp rv; // return value
    if (n == 1) { // base case

        rv.s = _________A[0]_________;

        rv.l = __________0__________; rv.r = __________0__________;        1
    } else { // inductive case
        int ls = n/2, rs = n-ls;

        rsTyp msrL = maxSumRngDC (_________ls_________, _________A_________);        1

        rsTyp msrR = maxSumRngDC (_________rs_________, _________A+ls_________);        1
        rsTyp msrM = maxSumRngMM (n, A, ls);
        if (msrL.s > msrR.s) {
            if (msrL.s > msrM.s) return msrL; else return msrM;
        } else {
            if (msrR.s > msrM.s) return msrR; else return msrM;
        }
    }
    return rv;
}
```

10. Examine the working of **`maxSumRngLR()`** carefully to understand how it can lead to a linear time solution; complete the code below to achieve that.

```
rsTyp maxSumRngLinear (int n, int A[n]) { // n>=1
// Linear time solution for identifying the max sum range
    int i; rsTyp rv;// return value
    rv.r = maxSumRngLR (

      _________n_________, _________A_________, _________1_________).

      _________r_________;        1
    rv.l = maxSumRngLR (

      _________n_________, _________A_________, _________-1_________).

      _________l_________;        1
    for (rv.s=0, i=rv.l; i<=rv.r; i++) rv.s += A[i];
    return rv;
}
```