ata Structures (201) Major Exam 1 Weight: 15% Time: 40 minutes

Name: ________________ ID ________________

SOLUTION

NOTE: This solution provide one possible solution for questions in this exam. It is possible that your answer is correct but is not similar to this solution. Please check with your instructor how your exam was graded.

Circle your instructor

Dr. Basit Qureshi Dr. Muhammad Akour Dr. Zahid Khan

Instructions:

- This exam contains four questions with multiple parts, on 5 sheets of paper (last sheet is scratch-sheet)
- Time allowed: 40 minutes
- Closed Book, Closed Notes.
- Use of Calculators and / or computing devices / smartphones etc is strictly prohibited.
- Answer the problems on the exam sheets only. No additional attachments would be accepted.
- **DO NOT write on the backside of a page/sheet**; the back of a page will NOT be graded.
- When the "time is over" is called, it is the students' responsibility to submit his exam to the invigilator. Submitting completed exam 3 minutes after the "time is over" will incur a penalty of **5 points**.

Few gentle reminders:

- If you get stuck on some problem for a long time, move on to the next one.
- The ordering of the problems is somewhat related to their relative difficulty. However, the order might be different for you!
- You should be better off by first reading all questions and answering them in the order of what you think is the easiest to the hardest problem.
- Keep the points distribution in mind when deciding how much time to spend on each problem.

START HERE

Q1 (a) [2 points]. Sam gives the run-time for an algorithm using function $f(x)$. Prove, for what values of n_0 and constant c, $f(x)$is $O(n^3)$.

$$f(n) = 2x^3 + 5x^2 + 12$$

$2n^3 + 5n^2 + 12 \leq cn^3$

$12 \leq cn^3 - 2n^3 - 5n^2$

$12 \leq (c-2)n^3 - 5n^2$

set $n_0 = 1$

$c = 20$

$12 \leq (20-2) \times 1 - 5 \times 1$

$12 \leq 18 - 5$

$12 \leq 13$

so $f(x)$ is $O(n^3)$ for $c = 20$, $n_0 = 1$

Q1 (b) [2 points]. Give the worst-case running time T(n) of the function Adder1 and provide the Big-Oh notation.

```
public int Adder1(int [][] A) {
  int sum=0;
  for (i = 0; i < n; i++){
    for (j = i; j < n; j++){

      sum = sum + A[i][j];
    }
  }
  return sum
}
```

1 operation

connected loop, you may use series $\frac{n(n+1)}{2}$
here i'm not using series.

$1 + n + n = 2n+1$

$n \times (2n+1) = 2n^2+n$

$n^2 \times 3$

return sum — 1

$T(n) = 1 + 1 + (2n+1) + (2n^2+n) + 3n^2 + 1$

$T(n) = 5n^2 + 3n + 4$

$O(n^2)$ – Quadratic. runtime.

*Please check with your instructor if your method for this estimation is differt.

Q1 (c) [2 points]. Describe the worst-case running time T(n) of the function Adder2 and provide the Big-Oh notation. Show/draw the recursion trace as necessary for this call:

```
Adder2(new int [] {1,2,3,4,5}, 0);

public int Adder2(int [] B, int x) {
  if(x == B.length)
    return 0;
  else
    return B[x] + Adder2(B, ++x);
}
```

1 operation.

3operation

Building Trace

$\boxed{\text{Adder 2}(B, 0)}$

$1 + \boxed{\text{Adder 2}(B, 1)}$

$2 + \boxed{\text{Adder 2}(B, 2)}$

$3 + \boxed{\text{Adder 2}(B, 3)}$

$4 + \boxed{\text{Adder 2}(B, 4)}$

$5 + \boxed{\text{Adder 2}(B, 5)}$

– 0
Base-case

$T(n) = n(3+1) + c$

$= 4n + c$

$O(n)$ – Linear run time

Q2 [3 points]. Write a method public **Node getSecondToLast(SList L)** that takes a Singly Linked List as a parameter. This method returns a reference to the node before the last-node in the list.

```
public Node getSecondToLast(SList L){
    Node tep = L.head;
    if(tep == null) return null
    if(tep.next = null) return null
    while(tep.next.next != null){
        tep = tep.next
    }
    return tep;
}
```

Q3 [3 points] Write a method **public int getMinElemStack(Stack S)** that takes an integer stack as a parameter. This method should return the minimum value in the stack S.

Hint: You may use additional **data structure(s) to ensure that the order of items in S is not changed.**

```
public int getMinElemStack(Stack S){
    Stack T = new Stack()   // assuming of type int.
    int min = S.top();
    int N = S.size();
    for(int i=0; i<N-1; i++){
        T.push(S.pop())
        if(S.top() < min){
            min = S.top()
        }
    }
    for(int i=0; i<N; i++){
        S.push(T.pop())
    }
    return min
}
```

Q4 [3 points] Assume an empty Queue Q of type int, is provided. Show/illustrate/Draw the contents of Q and provide the output (as needed) after each of these operations:

Operations	Output	Contents of Q
`Q.enqueue(3)`	—	3
`Q.enqueue(1)`	—	3, 1
`Q.first()`	3	3, 1
`Q.dequeue()`	3	1
`Q.isEmpty()`	false	1
`Q.dequeue()`	1	
`Q.size()`	0	
`Q.dequeue()`	Error or null	
`Q.enqueue(5)`	—	5
`Q.enqueue(4)`	—	5, 4
`Q.enqueue(3)`	—	5, 4, 3
`Q.size()`	3	5, 4, 3
`Q.enqueue(Q.dequeue())`	—	4, 3, 5
`Q.first()`	4	4, 3, 5