National University of Computer and Emerging Sciences

Department of Computer Science

CS 201 – Data Structures

Final Exam (Fall 2014)

**Instructors:** Adnan Waheed and Rabia Maqsood

*December 18, 2014*

**PART – B**

|  |  |
| --- | --- |
| **Total Marks: 26** | **Time Allowed: 90 minutes** |

**Instructions:**

1. Understanding the question is part of exam. NO QUERIES WILL BE ENTERTAINED.
2. Provide answers in the given space.
3. Use back side of each sheet, if required.
4. Write neat and clean.
5. Use permanent ink pens only.
6. Multiple answers will not be marked. Clearly mention which solution you want to be checked (if you provide multiple answers). In case of ambiguity, ZERO points will be assigned to the respective question(s).
7. Poor programming approaches will decrease your marks.

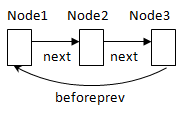
**Roll No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Question No.** | **1** | **2** | **3** | **4** | **5** | **Total** |
| **Marks** | ***04*** | ***05*** | ***05*** | ***06*** | ***06*** | ***26*** |
| **Obtained** |  |  |  |  |  |  |

**GOOD LUCK ☺**

|  |  |
| --- | --- |
| **Question 1:** | **Marks 04** |

Consider a special type of doubly linked list **doublyPrev**. Each node stores an integer and has two pointers *next* and *beforeprev* of type node. Pointer *next* points to the next node in the list whereas *beforeprev* points to the node immediately before the previous node as shown in the diagram below. Implement the delete function **Delete (int number)** for doublyPrev list.



**Solution:**

|  |  |
| --- | --- |
| **Question 2:** | **Marks 05** |

Write a piece of code which reverses the contents of a given stack with N number of elements. You cannot use any other data structure for any purpose.

*(HINT: recall the solution was discussed in ‘guest lecture’ session)*

**Solution:**

|  |  |
| --- | --- |
| **Question 3:** | **Marks 05** |

Professor Karim thinks he has discovered a remarkable property of BST. Suppose that the search for key k in a BST ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor Karim claims that any three keys ,  and  must satisfy . What is your opinion about professor’s claim? State your answer as TRUE or FALSE and support your answer in both cases with the help of a simple example.

**Solution:**

|  |  |
| --- | --- |
| **Question 4:** | **Marks 06** |

Suppose there is a long line of people at a check-out counter in a store. The manager decides to open up another check-out. To ensure fairness, he directs every alternate person (i.e. the second, fourth, sixth, etc.) to the new check-out, such that the new queue maintains the same relative order as the original queue.

Example: Let’s refer to the original queue as firstQ and the new queue as secondQ. Suppose firstQ contained customers: *c1, c2, c3, c4, c5, c6, c7*. After the split, firstQ contains customers: *c1, c3, c5, c7* and secondQ contains customers: *c2, c4, c6*.

Write a function ***splitCustomers (…)*** that “splits” off the alternate people from the original queue and puts these alternate people in the new (initially empty) queue. You are NOT writing a complete program; you are only writing the code that forms the firstQ queue and the secondQ queue from the original firstQ queue.

**void splitCustomers (Queue \*&firstQ, Queue \*&secondQ);**

You may assume that the following functions are already implemented for Queue class.

void enqueue (Customer c);

Customer dequeue ( );

int qSize ( );

bool isEmpty ( );

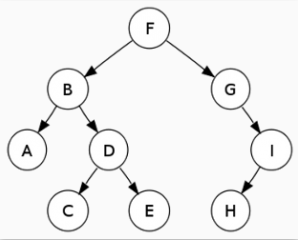
**Solution:**

|  |  |
| --- | --- |
| **Question 5:** | **Marks 06** |

Implement the following non-recursive function spiralTraversal which displays (integer) nodes of a binary tree at each level in a spiral (zig-zag) fashion. You may use any other data structure(s) *(if required)*, other than an array and a linked list *(no need to implement basic operations)*.

**void spiralTraversal (BST \*root);**

For example: F, B, G, I, D, A, C, E, H is the spiralTraversal sequence for the given BST.



**Solution:**