National University of Computer and Emerging Sciences, Lahore Campus

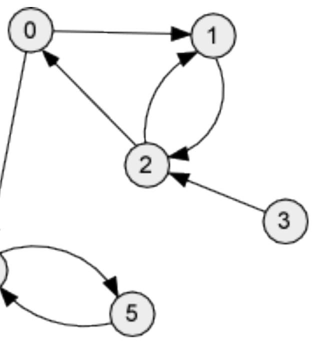
	Course:	Data Structures	Course Code:	CS 201
	Program:	BS (Computer Science)	Semester:	Fall 2017
	Duration:	180 Minutes	Total Marks:	80
	Paper Date:		Page(s):	
	Section:	N/A	Section:	
	Exam:	Final Exam (Retake)	Roll No:	

Instruction/Notes: Solve this exam on the answer sheet.

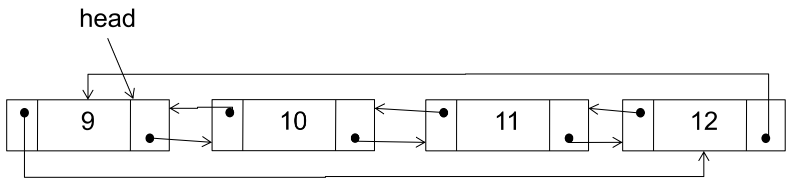
Q1. [20] Answer the following briefly, and explain your answer where required.

(a) H is an integer array containing a Max Heap with 20 elements in all, with the root element being at H[1], which is also the greatest element in the heap. On which indices of H may we find i) the 2nd highest element ii) the 3rd highest element? Explain why.

(b) What are the outputs of depth first traversal and breadth traversals as performed on the following graph? Start from node 0 and break ties in the numerical order.



(c) Performing the following sequence of operations on the linked list given below. Each operation is to be performed on the ouput of the previous one. Draw the resultant list after each operation.



a) head->next->next = head->prev;
b) head->prev->next = head->next;
c) head->next = head->prev->prev;
d) head->next = head;

(d) Give Big-Oh bounds on the worst-case and best-case running times of the following piece of code and exaplin your answers.

```
int multAdd(int A[], int n)
{
        if(n>=1000)
```

```
                {
                        int sum=0;
                        for(int i=0; i<=1000; i++)
                        {       if(A[i]<0)
                                {       for(int j=0;j<n;j++)
                                        {       sum=sum + j * A[i];
                                        }
                                }
                        }
               return sum;
                }
                else
                    return 0;
        }
```

Q2. [20] One common use of a stack is in implementing the undo and redo functionalites in such applications as browsers, text editors, board games, etc. In this question we ask you to implement a C++ class called UndoRedo. This class *cannot* use a separate stack class but must strore two arrays, and other appropriate attributes, to accomplish its functionality. The required class methods are: **undo()**, **redo()**, and **execute(const Operation&)**. You can assume that global functions called **exec(const Operation&)** and **cancel(const Operation&)** are available to run and cancel the operations passed to them. A fully-functional class called Operation, containing all necessary operators, already exists.
The methods are described below:

- **undo :** makes appropriate changes to the undo and redo stacks and cancels the undone operation by calling cancel.
- **redo:** makes appropriate changes to the redo and undo stacks and uses exec to execute the redone operation
- **execute(Operation&):** makes the input operation available for undo and executes it using exec.

Further notes and requirements:

- At any given time the undo stack contains *at most* 256 most recently executed or redone operations.
- At any given time the redo stack contains *at most* 256 most recently undone operations.
- Each call to each of these methods should take exactly O(1) time.
- You may assume that exec and cancel are O(1) functions.

Q3. [15] Add a function called updateKey to the class MinHeap; the skeleton class is given below. The inputs to update key are the index of the key to be updated, and the new value. You must implement any other function you may want to use inside updateKey.

```
class MinHeap{
            int * H; //heap keys, starting from H[1]
            int count; //number of elemets in H
            int size; //total size of H
            …
        public:
                void updateKey(int index, int newVal);

}
```

Q4. [15] Write a recursive function which reverses the keys of a singly linked list pairwise. For example, if the list is: 1 → 4 → 8 → 3 → 9 → 2 → null, after the function has been executed it becomes: 4 → 1 → 3 → 8 → 2 → 9 → null. You can assume that the list always has even number of elements. **Note**: iterative solution is not acceptable.

Q5. [10] Perform the following insertions on an AVL tree starting from an empty tree. Show the tree after each insertion.

0, 6, 20, 36, and 88