

# CENG 707 – Data Structures and Algorithms

## Midterm Exam #2 08/12/2011

**NAME SURNAME, ID:** \_\_\_\_\_

**Duration:** 60 minutes

**Exam:**

- This is a **closed book, closed notes** exam. The use of any reference material is strictly forbidden.

**About the exam questions:**

- The points assigned for each question are shown in parentheses next to the question number.

**This exam consists of 5 pages including this page. Check that you have them all.**

**GOOD LUCK!**

---

**Question 1**

**Question 2**

**Question 3**

**Total:**

**1** (30 points)



Fill in the blank code segments in the following implementation of a **doubly linked list of integers**, so that it is a correct implementation of doubly linked lists. The comments in the code relate to the code segment immediately below the comment.

```
typedef struct Node
{
    int element;
    _____ next;
    _____ prev;
} Node;

/* insert function inserts newData at the end of the
list */
Node* insert(Node *head, int newData)
{
    /* if list is empty create and return new head */
    if (head == NULL)
    {
        _____
        _____
        _____
    }
    /* go to the end of the list*/
    Node *tmp = head;
    while (tmp->next!=NULL)
        _____

    /* create new node and arrange the links */

    tmp->next = (Node *)malloc(sizeof(Node));
    _____
    _____
    _____

    return head;
}
```

```

/* find function finds the first occurrence of data in
the list and returns a pointer to its node. It returns
NULL if data is not in the list*/
Node* find(Node *head, int data)
{
    _____

    while ( _____ )
        head = head->next;
    return head;
}
/* delete function deletes the first occurrence of
data from the list and does nothing if data is not in
the list*/
Node* delete(Node *head, int data)
{
    Node *tmp;
    tmp = find(head,data);
    /* do nothing if empty or not in list
    if (tmp==NULL)    return head;
    /* delete the first element */
    if (tmp==head)
    {
        _____
        _____
        _____
    }
    else
    /* delete an element which is not the first */
    {
        _____
        _____
        _____
        _____
    }
    return head;
}

```

**2** (40 points)



Write a function using a stack of characters to check whether the opening and closing parenthesis in an arithmetic expression are balanced. Assume that the following functions of stack are available to you:

createStack() : creates an empty stack  
push(char a) : pushes a character on the stack  
pop() : pops the top character from the stack  
peek() : returns the top character without deleting it  
isEmpty() : checks whether the stack is empty

An example usage of the above functions are given below:

```
Stack myStack = createStack();  
myStack.push(5);  
if (!myStack.isEmpty()) myStack.pop();
```

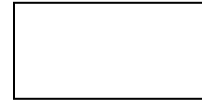
The prototype of the function you are going to write is given below. It should return **true** if the parentheses are balanced and **false** otherwise.

boolean checkParenthesis (string expression)

. Assume that the string class has the following functions:

length() : returns the length of the string  
charAt(int ind) : returns the character at index ind, The first character is at index 0.

**3** (30 points)



(a) 15pts

Write a recursive function to find the number of leaves in a binary tree. Assume that you have the following node structure:

```
struct Node {  
    int element;  
    struct Node *left;  
    struct Node *right;  
};
```

(b) 15pts

Write a recursive function to check whether a given binary tree is a **binary search tree** or not.