Philadelphia University
Faculty of Engineering

Marking Scheme

Exam Paper

BSc CE

Algorithms and Data Structures (630231)

Second Exam | First semester | Date: 26/12/2010

Section 1

Weighting 15% of the module total

Lecturer:	Dr. Qadri Hamarsheh
Coordinator:	Dr. Qadri Hamarsheh
Internal Examiner:	Dr. Ali Al-Khawaldeh

Marking Scheme

Algorithms and Data Structures (630231)

The presented exam questions are organized to overcome course material through 5 questions.
The *all questions* are compulsory requested to be answered.

Marking Assignments

Question 1 This question is attributed with 3 marks if answered properly; the answers are as following:

1. In a doubly linked list, every node contains the address of the next node except for the ____ node.
   a) middle
   b) first
   c) last
   d) second to last
2. A queue is a ____ data structure.
   a) Last In First Out
   b) Last In Last Out
   c) First In Last Out
   d) First In First Out
3. The queue operation ____ returns the first element of the queue
   a) front
   b) tail
   c) delete
   d) insert

Question 2 This question is attributed with 3 marks if answered properly; the answers are as following:

Question 2-a

Solution

Receive: An RPN expression.
Return: A stack whose top element is the value of RPN expression (unless an error occurred).

1. *Initialize an empty stack.*
2. Repeat the following until the end of the expression is encountered:
   a. *Get next token* (constant, variable, arithmetic operator) in the RPN expression.
   b. If token is an *operand, push it onto the stack.*
      If it is an *operator*, then
      (i) *Pop top two values from the stack.*
      If stack does not contain two items, error due to a malformed RPN
      Evaluation terminated
      (ii) *Apply the operator* to these two values.
      (iii) *Push the resulting value back onto the stac*k.
3. When the end of expression encountered, its value is on top of the stack
   (and, in fact, must be the only value in the stack).

Question 2-b

Solution

The value is: 3, the infix of this is: ((5 - 1) * 3) / (3 – 1) * 2).

Question 3 This question is attributed with 1.5 marks if answered properly. The complete code for this question

```
stack<string> s;
s.push("hello");
s.size()
```

Question 4 This question is attributed with 3.5 marks if answered properly. The complete code for this question

```
void print_stack_inorder(Stack my_stack)
{
Stack tempStack;
int temp_val;
int current_min;
while(!my_stack.isEmpty()) { // Do nothing w. empty stack
current_min = my_stack.top();                                    (1 mark)
// Pop all values off of my_stack and push onto tempStack.
while(!my_stack.isEmpty()) {
temp_val = my_stack.pop();
tempStack.push(temp_val);
```

```
            // Keep track of the minimum value.
            if (temp_val < current_min)
            current_min = temp_val;
            }                                                                    (1 mark)
            // Pop all values off of tempStack and push onto my_stack.
            while(!tempStack.isEmpty()) {
            temp_val = tempStack.pop();
            // Print out the current min and don't push back on.
            if (temp_val == current_min)
            cout << current_min << " " ;
            else
            my_stack.push(temp_val);
            }
            }
            cout << endl;
            }                                                              (1.5 marks)
```

Question 5 This question is attributed with 4 marks if answered properly. The complete code for this question:

```
template <class Type>
void orderedLinkedList<Type>::mergeLists(orderedLinkedList<Type> &list1,
                                         orderedLinkedList<Type> &list2)
{
      nodeType<Type> *lastSmall;    //pointer to the last node of the merged list.
      nodeType<Type> *first1 = list1.first;
      nodeType<Type> *first2 = list2.first;
      count = list1.count + list2.count;
      if (list1.first == NULL)   //first sublist is empty
      {
            first = list2.first;    list2.first = NULL;  count = list2.count;
      }
      else if (list2.first == NULL)   // second sublist is empty
    {
       first = list1.first; list1.first = NULL; count = list1.count;
    }                                                                  (1.5 marks)
    else
    {
       if (first1->info < first2->info) //Compare first nodes
       {
           first = first1;first1 = first1->link; lastSmall = first;
       }
       else
       {
           first = first2; first2 = first2->link; lastSmall = first;
       }

       while (first1 != NULL && first2 != NULL)
       {
           if (first1->info < first2->info)
           {
               lastSmall->link = first1; lastSmall = lastSmall->link;
               first1 = first1->link;
           }
           else
           {
               lastSmall->link = first2; lastSmall = lastSmall->link;
               first2 = first2->link;
           }
        } //end while                                                        (1.5 marks)
       if (first1 == NULL)              //first sublist exhausted first
           lastSmall->link = first2;
       else                             //second sublist exhausted first
            lastSmall->link = first1;
       list1.first = NULL;  list1.last = NULL;
       list2.first = NULL;  list2.last = NULL;
       count = list1.count + list2.count;
      }
}                                                                          (1 mark)
```