Philadelphia University
Faculty of Engineering

Marking Scheme

Exam Paper

BSc CE

Algorithms and Data Structures (630231)

Final Exam First semester Date: 26/01/2011

Section 1

Weighting 50% 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 8 questions.
The *all questions* are compulsory requested to be answered.

Marking Assignments

Question 1 This question is attributed with 10 marks if answered properly; the answers are as following:

1. The three fundamental stages a program goes through are: development, use, and ____.
   a. implementation
   b. maintenance
   c. analysis
   d. requirements gathering
2. In a(n) ____ copy, two or more pointers of the same type point to the same memory.
   a. indirect
   b. direct
   c. deep
   d. shallow
3. In an array list the time complexity of the `remove` function is identical to the time complexity of the ____ function.
   a. insert
   b. isEmpty
   c. isFull
   d. maxListSize
4. If the data needs to be processed in a First In First Out (FIFO) manner, we typically use a(n) ____.
   a. stack
   b. queue
   c. map
   d. hash table
5. Because initially the list is empty, the pointer `first` must be initialized to _____.
   a. NULL
   b. EMPTY
   c. NIL
   d. NOP
6. A recursive function executes ____ its iterative counterpart.
   a. more slowly than
   b. more quickly than
   c. at the same speed as
   d. proportionately to
7. Because all the elements of a stack are of the same type, you can use a(n) ____ to implement a stack.
   a. struct
   b. array
   c. record
   d. class
8. To speed up item insertion and deletion in a data set, use ____.
   a. arrays
   b. linked lists
   c. classes
   d. ADTs
9. The height of a perfectly balanced binary tree with n nodes is ____.
   a. log$n$
   b. n^2
   c. $n$log$n$
   d. $\log_2 n$
10. The ____ provides class templates to process lists (contiguous or linked), stacks, and queues.
    a. STL
    b. ITL
    c. CTL
    d. ADL

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

a)	`for (int j = 0; j < n; ++j)`
	`{`
	`for (int i = 0; i < j; ++i)`
	`a = a + b;`
	`for (int i = 0; i < n; ++i)`
	`{`
	`c = b + c;`
	`cin >> b;`
	`}}`
		$O(n^2)$

b)	`for (int i = 0; i < 999999; ++i) {`
	`for (int j = 0; j < n; ++j) {`
	`a[i]= a[i] + b + c;`
	`}`
	`for (int k = 0; k < n; ++k) {`
	`cout << i;`
	`}}`
		$O(n)$

c)	`for (int i = 0; i < 999999; ++i) {`
	`for (int j = 0; j < n; ++j) {`
	`a[i]= a[i] + b + c;`
	`}`
	`for (int k = 0; k < n; ++k) {`
	`cout << i;`
	`}}`
		$O(n)$

Question 3 This question is attributed with 7 marks if answered properly; the answers are as following

Question 3.a

Array: fast operations; no pointer following; no slowdown from new/delete operations; takes less space because there are no pointers to next or previous elements.

Linked list: flexible; we don't have to decide in advance how big the queue can get. **(2 marks)**

Question 3.b

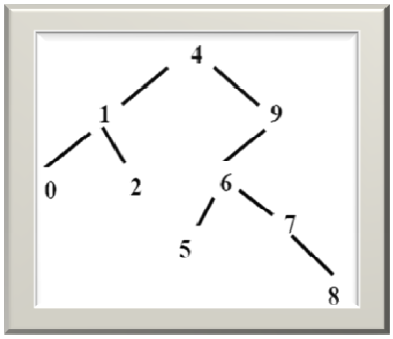
Replace the node being deleted with the leftmost child of the right subtree. You could also replace it with the rightmost child of the left subtree. **(1 mark)**

Question 3.c

Maximum number of comparisons required: **(2 marks)**

items	Sequential Search algorithm	Binary Search algorithm
3	3	2
1,023	1,023	10
65,535	65,535	16

Question 3.d **(1 mark)**



Question 3.e **(1 mark)**

4 1 0 2 9 6 5 7 8

Question 4 This question is attributed with 5 marks if answered properly; the answers are as following

binaryTreeType
+isEmpty() const: bool +inorderTraversal() const: void +preorderTraversal() const: void +postorderTraversal() const: void +treeHeight() const: int +treeNodeCount() const: int +treeLeavesCount() const: int +binaryTreeType(); +binaryTreeType(const binaryTreeType<elemType>& otherTree); +~binaryTreeType(); +operator= (const binaryTreeType<elemType>&): const binaryTreeType<elemType>& +destroyTree():void
#binaryTreeNode<elemType> *root;
-copyTree(binaryTreeNode<elemType>* &copiedTreeRoot,binaryTreeNode<elemType>* otherTreeRoot): void -destroy(binaryTreeNode<elemType>* &p): void -inorder(binaryTreeNode<elemType> *p) const: void -preorder(binaryTreeNode<elemType> *p) const: void -postorder(binaryTreeNode<elemType> *p) const: void -height(binaryTreeNode<elemType> *p) const: int -max(int x, int y) const: int -nodeCount(binaryTreeNode<elemType> *p) const: int -leavesCount(binaryTreeNode<elemType> *p) const: int

Question 5 This question is attributed with 6 marks if answered properly; the complete code for this question:

Question 5.a

```
int compare_linked_lists(struct node *q, struct node *r)
{
   int flag;
   if((q==NULL ) && (r==NULL))
       {
       flag=1;
       }                                                          (1 mark)
   else
   {
       if(q==NULL || r==NULL)
       {
       flag=0;
       }                                                          (1 mark)
       if(q->data!=r->data)
       {
       flag=0;
       }                                                          (1 mark)
       else
       {
       return compare_linked_lists(q->link,r->link);
       }                                                          (1 mark)
   }
   return(flag);
}
```

Question 5.b (2 marks)

```
Node *Temp2 = new Node;
Temp2-> Value = “6”;
Temp2->Next = Temp1->Next;
Temp1->Next = Temp2;
```

Question 6 This question is attributed with 5 marks if answered properly; the complete code for this question:

```
#include <stack>
#include <iostream>
using namespace std;
int main()
{
unsigned number, // the number to be converted
remainder; // remainder when number is divided by 2
Stack stackOfRemainders; // stack of remainders                   (1 mark)
char response; // user response
do
{
cout << "Enter positive integer to convert: ";
cin >> number;
while (number != 0)
{
remainder = number % 2;
stackOfRemainders.push(remainder);
number /= 2;
}                                                                 (2 marks)
cout << "Base-two representation: ";
while (!stackOfRemainders.empty() )
{
remainder = stackOfRemainders.top();
stackOfRemainders.pop();
cout << remainder;
}
cout << endl;
cout << "\nMore (Y or N)? ";
cin >> response;
}
while (response == 'Y' || response == 'y');
}                                                                 (2 marks)
```

Question 7 This question is attributed with 9 marks if answered properly; the complete code for this question:

Question 7.a

```
Queue::Queue()
{
      head = NULL;
      tail = NULL;
      size = 0;
}                                                                    (1 mark)
void Queue::enqueue(int value)
{
      Node *temp = new Node(value);
      if (isEmpty())
            head = tail = temp;                                      (1 mark)
      else { // there is at least one element in the queue
            tail->next = temp;
            temp->prev = tail;
            tail = temp;
            }
      size++;
}                                                                    (2 marks)
```

Question 7.b

```
void Queue::printQueue(bool chron) const
{
      Node *temp;
      if (chron)
            { // print from head to tail
            temp = head;
            while (temp)
            {
            cout << temp->info << " ";
            temp = temp->next;
            }
            }                                                        (2.5 marks)
     else
     { // print from tail to head
            temp = tail;
            while (temp)
            {
            cout << temp->info << " ";
            temp = temp->prev;
            }}}                                                      (2.5 marks)
```

Question 8 This question is attributed with 5 marks if answered properly; the complete code for this question:

```
int main()
{
   TreeNode *root;
   root = NULL;
char item;
cout << ("\n\nEnter a char to be inserted, or q to end.\n");
cout << ("?  ");
cin >> item;                                                         (1 mark)
   while (item != 'q') {
       if (treeContains(root,item)) {
          cout << "\nThat item is already in the tree.\n";
       }
       else {
          treeInsert(root,item);  // Add user's char to the tree.
          cout << "\nThe tree contains " << countNodes(root) << " items.\n";
          cout << "\nContents of tree:\n\n";
          treeList(root);
       }
cout << ("\n\nEnter a char to be inserted, or press return to end.\n");
cout << ("?  ");
cin >> item;
   }                                                                 (4 marks)
   cout << "\n\nExiting program.\n\n";
return 0;}
```