Level 1 – Semester 2 Examination

DATA STRUCTURES AND ALGORITHMS 1

Time Allowed: 3 hours

INSTRUCTIONS TO CANDIDATES

1. This paper contains 4 question on 6 Pages.
2. The total marks obtainable for this examination is 100. The marks assigned for each section is included in square brackets.
3. This examination accounts for 70% of the module assessment.
4. This is an closed book examination.
5. Answer ALL questions.

Question 1 [Total marks allocated: 25 marks]

(a) List two (2) differences between recursion and iteration?

[2 Marks]

(b) Implement the selection sort algorithm using c.

[4 Marks]

(c) Briefly explain two (2) advantages and one (1) disadvantages of selection sort.

[3 Marks]

(d) Using the Selection Sort algorithm, sort the numbers 8, 2, 5, 9, 7, 8. State the steps used in the sorting process.

[4 Marks]

(e) What are the major steps of the bubble sort algorithm?

[2 Marks]

(f) Compare Best, Worst, Average time complexity of Insertion sort, Selection sort, Bubble sort.

[3 Marks]

(g) Using the insertion algorithm, sort the numbers 8, 2, 5, 9, 7, 8. State the Steps used in the sorting process.

[7 Marks]

Question 2 [Total marks allocated: 25 marks]

(a) Briefly describe the following data structures:

I. Stack
II. Queue
III. Linked list

[6 marks]

(b) Write a c program to implement a stack using an array by following steps. Your functions should be efficient.

I. Define a variables MAX_SIZE and TOP to store the size of the stack array and the index of the top element, respectively. Initialize TOP to -1.
II. Define array STACK_DATA to store the stack array of int.
III. A function to test whether the stack is empty or not.
IV. A function to test whether the stack is full or not.
V. A function to insert an int onto the stack (Your method should for overflow of the stack).
VI. A function to remove the item at top and returns the removed item (Your method should check for underflow of the stack).

[2 x 6 marks]

(c) Make a simple Function empty_Stack using Recursion to pop all the items from the stack.

[7 marks]

Question 3 [Total marks allocated: 25 marks]

(a) Compare Arrays and LinkedList data structures.

[3 Marks]

(b) Provide a real-world example in which the queue data structure can be applied.

[2 Marks]

(c) Here is a code segment of Queue data structure example implemented using a linked list:

```
struct node{
    int data;
    struct node *next;
};
struct node *front = NULL;
struct node *rear = NULL;
```

I. create isEmpty() function to checks if the queue is empty.

[3 Marks]

II. Create peek() function to get the element at the front of the queue without removing it.

[3 Marks]

III. Create enqueue(int data) function to add (store) an item to the queue.

[4 Marks]

(d) Consider that you have an integer-typed queue Q and an integer-typed Stack S. graphically describe the statuses of S and Q after each of the following operations:

S.push (4)
s.push (10)
Q.enqueue (6)
Q.enqueue (8)
x = S.pop()
S.push (2)
Q.enqueue(x)
Y = Q.dequeue()
S.push(x)
S.push(y)

[1 x 10 Marks]

Question 4 [Total marks allocated: 25 marks]

(a) Briefly describe the concepts of static and dynamic memory allocations.

[2 Marks]

(b) Consider the following linked list with four (4) nodes as shown in Figure l. The first node is the head, and the last node is the tail. Each node has *data* field to store an element, *next* field to the next node in the list, and *prev* field to point to the previous node in the list.

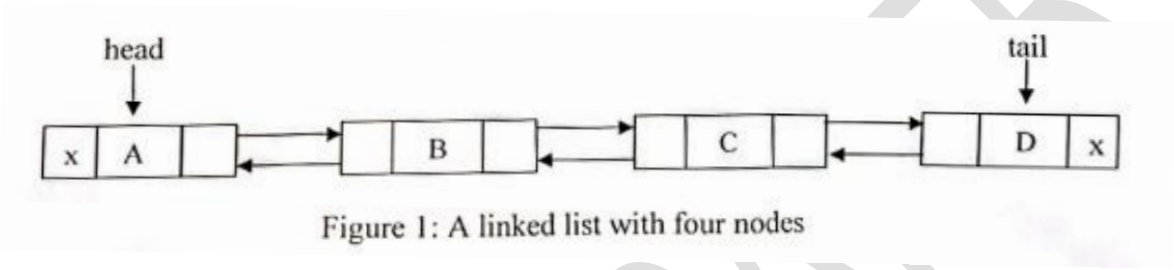

Figure 1: A linked list with four nodes

For each of the code fragments shown draw the Linked list. The list is restored to its initial state before each fragment executed.

I. head.next.prev = null;
   head = head.next;

II. tail.prev.data = head.next.data;

III. head.next.next.next.data = tail.prev.prev.prev.data;

IV. head.data = head.next.prev.data;

V. tail.prev.next = null;
   tail = tail.prev;

[2 x 5 Marks]

(c) Define the following terms.

I. Time complexity
II. Space complexity

[2 Marks]

(d) Find the worst-case time complexity of the following c code segment in terms of variable n. Explain the steps of the calculation of time complexity.

```c
void testFunction(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 10; j++) {
            for (int k = 0; k < n; k++) {
                for (int m = 0; m < 20; m++) {
                    printf("*");
                }
            }
        }
    }
}
```