Short questions

1. What are the two important things one must remember when designing a recursive solution? Is there any advantage of using recursion? [1]
2. Write two difference between queue and priority queue. [1]
3. Suppose you run a O(log n) algorithm with an input size of 1000, how many operations does the algorithm require for 1000 input size? When you double the input size to 2000, does the algorithm now requires double number of operations? What is your best guess for the number of operations taken by the algorithm for each of the input size? [1]

1. Write the big-Oh of the function below. [1]

void funcABC(int n) {

for (int i = 0; i <= n/2; ++i){

for (int j = 0; j < =n/4; ++j){}

}

for (int l = 0; l < =n/8; ++l){}

System.out.println(i\*i);

}

1. For which operation a doubly linked list would result in O(1) but a singly linked list would result in O(n)? Write the pseudo code of the function. [1]
2. Consider the following code,

ArrayQueue<Integer> q = new ArrayQueue<Integer>( );

q.add(3); q.add(1);

System.out.println(q.remove( ));

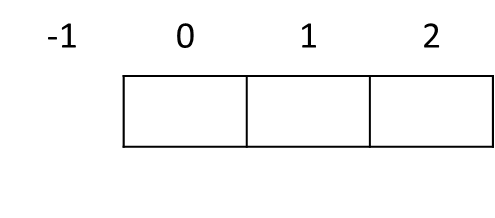
q.add(2); q.add(6);

q.add(5); q.add(9);

q.add(7); q.add(8);

System.out.println(q.remove( ));

Suppose that q is initially empty and represented by a circular array of size 6. Draw the state of the q with rear and front pointers after the above code executed. [1]

1. Consider a priority implemented as a circular array of size 3. When queue is empty, front = rear = -1. Draw changes occur to the priority queue after each of the following operations. [1]
   1. Enqueue(1)
   2. Enqueue(7)
   3. Enqueue(6)
   4. Dequeue()
   5. Dequeue()
   6. Enqueue(3)
   7. Enqueue(5)
2. Briefly describe a circular singly linked list. No need to write code or pseudo code. [1]

Programming questions

Note: write the Java code to answer the following questions.

1. Write a function called insertAt for a dynamic array class. Suppose there is a variable called ‘size’ which holds the number of items present in the array. The definition of the function is,

Public void insertAt (int index, int value){ // your code here }.

The insertAt function takes an index and a data value as input, and does the following,

* If the index is less than size, then it simply updates the value at the given index using the given value and returns.
* If the index is greater than size, then it fills the array with the data value from the index equal to size till the provided index. If the index is greater than length of the array, then it first resizes the array to size equal to index+2. For example, if an array has size 3 and length 5, it looks like this,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 3 | 12 | 4 |  |  |

Then, after a call insertAt(6, 22), it becomes,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 12 | 4 | 22 | 22 | 22 | 22 |  |

Note, the size has become 7 and length has become 8 (index+2). [4]

1. Write a recursive function that returns the sum of the digits of a positive integer. For example: SumOfDigits(int x) when x=163 will return 10 because 1+6+3=10. Hint: The modulo operator (%) is used to compute the remainder of an integer division, and the division operator (/) is used to compute the quotient of an integer division. Use the following function definition. [4]

public int SumOfDigits(int x){ // your code here }

1. Write an efficient function (isEvenNodes) which returns true if a linked list contains even number of nodes, otherwise returns false if the list contains odd number of nodes. Assume the following: the linked list’s head node is provided as a parameter to the function, the Node class has a data field of Integer type and a next variable of Node type, and the next of last node is null. Use the following function definition. [4]

public bool isEvenNodes(Node head){ // your code here }