Data Structures
Test 3
Instructor: Vickram Sawh

Topics: Recursion, Linked Lists, Generic Classes, Array implementation of an ordered list, Collection classes, Linked List implementation of an ordered list, Iterators and ListIterators, Stacks, Queues, Searching, the efficiency of Algorithms, Sorting, Hashing, Trees

---

Question 1.

Which of the following real world situations represents a **Stack** or a **Queue**. (Write the word **stack**, or **queue** in the blanks to the right of the listing)

1. The line of cars at a drive-up window of a fast food restaurant. _____
2. The paper in a tray of the printer. _____
3. The airplanes entering the airspace at an airport _____
4. The torpedoes loaded in a torpedo tube of a submarine _____
5. Plates in the sink, placed on top of each other. _____

Question 2.

In the space below, mention the operations allowable on a queue. Also explain what each operation does:

_____
_____
_____
_____
_____
_____

Question 3.

Part a) Clearly indicate the difference between a **Queue** and a **Priority Queue**.

_____
_____
_____
_____
_____
_____
_____

Part b) Give one (non-computer related) real world example of a **Priority Queue**:

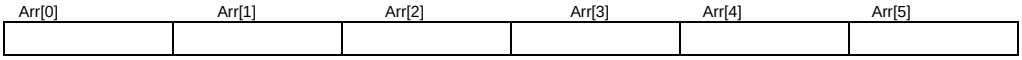
_____
_____

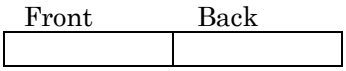
Question 4.

In implementing a queue using an array, it is necessary to avoid shifting of elements, due to adding/removing elements. It is also necessary to maintain two pointers: **Front** and **Back**. Both the **Front** and **Back** will be moving as elements are added / deleted, and the data items

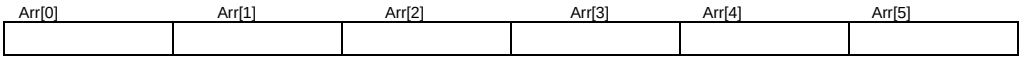
in the array will "wrap around" the array. In this question, you may assume that the **enqueue( )** method increases the **Back** pointer by one before adding the item into the array.

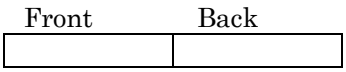
In the space below, mention how the array, named **Arr**, is initialized. Also, indicate the values stored in **Front** and **Back**:



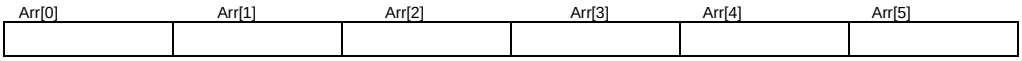


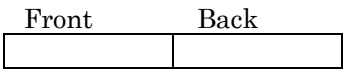
In the space below, mention what the array will look like (including the values for **Front** and **Back**), once the data item **12** is enqueued.



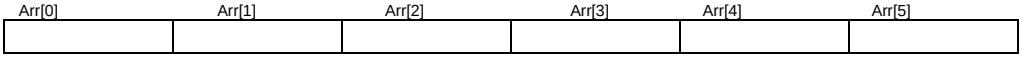


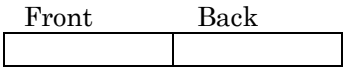
In the space below, mention what the array will look like (including the values for **Front** and **Back**), when the 2nd item of value **-3** is enqueued.



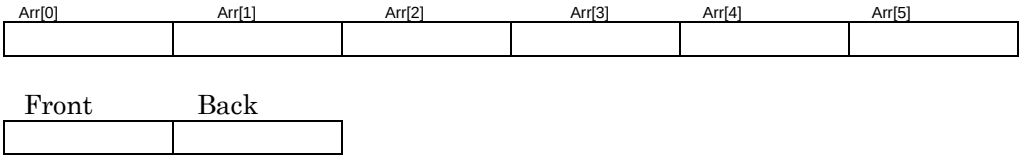


In the space below, mention what the array will look like (including the values for **Front** and **Back**), when the 3rd item of value **48** is enqueued.





In the space below, mention what the array will look like (including the values for **Front** and **Back**), when a dequeue happens:



In the space below write the **isFull( )** method, which returns **true** if the queue is full. (Of course, this method returns a **boolean**)

Question 5.

The algorithm **a1** has an efficiency of **O(n^2)**. The algorithm **a2** has an efficiency of **O($\log_2$ n)**. The algorithm **a3** has an efficiency of **O(1)**. Finally, algorithm **a4** has an efficiency of **O(n)**. All 4 algorithms are applied to the same data set, which contains a very large number of elements. In the space below, arrange the 4 algorithms in order according to their likely completion time. (Place the slower ones on the left of the faster ones)

_____

slowest **fastest**

Question 6.

Consider the definition of the efficiency of algorithms:

> **f(n)** is O(**g(n)**) if there exists a **c** and **N**, such that **f(n)** ≤ **c**.**g(n)** for all **n** ≥ **N**. (whereby **c** and **N** are both positive, and **c** is a real number and **N** is an integer)

Assuming the f(n) equals $\mathbf{3x^2+x}$ and g(n) equals $\mathbf{x^2}$. For what possible values of **c** and **N**, will the definition show that **f(n)** is O(**g(n)**) ?

Question 7.

An array is to be sorted using **insertion sorting**. The insertion sort algorithm is shown below:

```
public static void InsertionSort( int Arr[] )
{
        for (int unsorted=1;unsorted<Arr.length; unsorted++)
        {
                int NextItem=Arr[unsorted];
                int Loc=unsorted;
                for(;(Loc>0) && (Arr[Loc-1]>NextItem);--Loc)
                  Arr[Loc]=Arr[Loc-1];

                Arr[Loc]=NextItem;
        }  //Line 1
}
```

Assume that the following is the state of the array **Arr** at the moment that **InsertionSort( )** is called:

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]
8	19	7	10	-1	14

In the space of below, give the state of the array, when **Line 1** is reached for the <u>first</u> time:

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>second</u> time:

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>third</u> time:

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>fourth</u> time:

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]

Question 8.

An array is to be sorted using **selection sorting**. The selection sort algorithm is shown below:

```
public static void SelectionSort( int A[] )
{
        for (int i = 0 ; i < A.length-1 ; i++ )
        {
                int minIndex=i;
                int minValue=A[i];
                for ( int j=i+1 ; j <= A.length-1 ; j++ )
                {
                        if ( minValue > A[j] )
                        {
                                minValue=A[j];
                                minIndex=j;
                        }
                }

                int temp=A[i];
                A[i]=A[minIndex];
                A[minIndex]=temp;
        }    //Line 1
}
```

Assume that the following is the state of the array **A** at the moment that **SelectionSort( )** is called:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]
8	19	7	10	-1	14

In the space of below, give the state of the array, when **Line 1** is reached for the <u>first</u> time:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>second</u> time:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>third</u> time:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

In the space of below, give the state of the array, when **Line 1** is reached for the <u>fourth</u> time:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

Question 9.

Consider the following **merge sort** algorithm shown below:

```
public static void MergeSort( int Ar[], int first, int last)
{
        if (first < last)
        {
                int mid=(first + last) /2;

                MergeSort(Ar, first, mid);
                MergeSort(Ar,mid+1,last);

                Merge(Ar,first,mid, last);    //Line 1
        }
}


private static void Merge( int Ar[], int first, int mid, int last)
{
        int[] tempArray=new int[  Ar.length ];

        int first1=first;
        int last1=mid;
        int first2=mid+1;
        int last2=last;

        int index=first1;
        for(;(first1<=last1) && (first2<=last2); ++index)
        {
                if (Ar[first1] < Ar[first2])
                {
                        tempArray[ index ]=Ar[first1];
                        ++first1;
                }
                else
                {
                        tempArray[index]=Ar[first2];
                        ++first2;
                }
        }


        for(; first1 <= last1; ++first1,++index)
                tempArray[index]=Ar[first1];


        for (;first2 <=last2;++first2,++index)
                tempArray[index]=Ar[first2];

        for (index=first;index<=last;++index)
                Ar[index]=tempArray[index];
}
```

a) In the space below, explain in **30 words or less**, what the purpose is of the method **Merge( )**. (Do not explain how the algorithm does in a step by step manner!) Explain what its parameters represent, and what the final outcome of **Merge** is.

b) Assuming that the following array is passed to the **MergeSort( )** method:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]
9	5	8	10	-1	14

In the space below, give the state of the array after the very first call to **Merge( )** has been completed: (in other words, immediately after line 1 has been completed for the very first time)

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

In the space below, give the state of the array immediately after the second call to **Merge( )** has been completed:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

In the space below, give the state of the array immediately after the third call to **Merge( )** has been completed:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

Question 10.

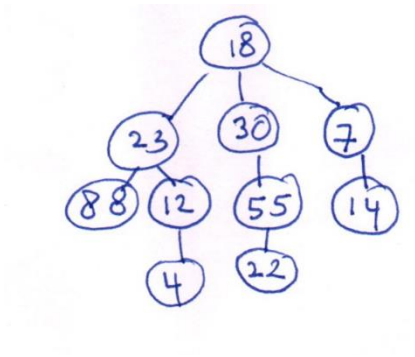
Assume that an array named **a** is to be processed, using the following algorithm:

```
for (k goes from 1 to a.length-1)
   for( i goes from 0 to a.length-k-1 )
        process a[i];
```

You may assume that upon each iteration of the inner loop, the processing statement takes the same amount of time. In the space below, prove that the efficiency of this algorithm is $O(m^2)$, with m being the number of cells in the array. You must show the exact derivation:

Question 11.

Consider the following tree:



Part a) Give <u>two</u> reasons why this is a general tree, NOT a Binary Search tree.

1. _____

2. _____

Part b)

What is the **height** of the tree? _____

What is the **level** of the node containing the value 4? _____

What are the values in the nodes which are **leaves**? _____

Which are the values of the nodes which are **descendants** of the node containing the value 18?
_____

Part c)

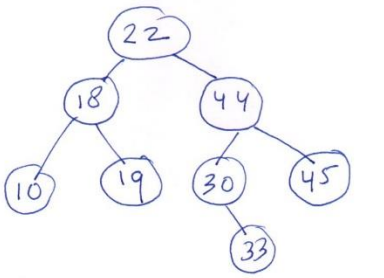
Assuming that each node contains integers as data. Also assume that the tree is implemented using linked lists. In the space below, write the code to describe a node of the tree above:

Part d)

Assuming that the tree above is implemented using a linked list, given the description of the node in the previous question. In the space below, draw how the above tree, will be represented in the memory of the computer:

Question 12.

Given the following Binary Search Tree: (fig 1)



In the space below, re-draw the tree after adding a node containing **36**:

Assuming the tree depicted in fig1, redraw the tree in the space below, after adding the value **21**:

Assuming the tree depicted in fig1, redraw the tree after deleting the node containing the value **30**:

Assuming the tree depicted in fig1, give the two possible ways the tree can look, after deleting the node containing **22**:

Question 13.

See the Binary Search Tree shown as fig 1 in the previous question. List the values in the nodes, if the BST is printed **post-order**:

List the values in the nodes, if the BST is printed **in-order**:

List the values in the nodes, if the BST is printed **pre-order**: