Part 1 – Java Review

This part went over the basics of Java, and the focus was mostly on arrays. There will be no questions that involve reading from and writing to files.

Short Answer

1. Given this programming snippet, what will print out at the end?

```java
final int a = 10;
for(int i=0;i<a;i+=2)
{
    for(int j=i;j<a;j++)
    {
        System.out.print("*");  
    }
    System.out.println();
}

 **********
 **********
 ******
 ****
 **
```

2. Will this piece of code give an error? If so why? If not what will it print out?

```java
int[] a = {1,2,3,4,5,6,7,8,9,10};
for(int i=1;i<5;i++)
{
    System.out.println(a[i]);
    System.out.println(a[i*2]);
}
```

2
3
3
5
4
7
5
9
3. Given this programming snippet, what will print out at the end?

```java
int[][] a = new int[5][5];
for (int i=0; i<a.length; i++)
{
    for (int j=0; j<a[i].length; j++)
    {
        a[i][j] = i+j;
    }
}
for (int i=1; i<a.length; i+=2)
{
    for (int j=a.length-1; j>=0; j-=2)
    {
        System.out.print(a[i][j]+" ");
    }
    System.out.println();
}
```

5 3 1
7 5 3
4. Write a method that takes in an array of integers and will print out any values in array that are divisible by 7

```java
public void printOut7s(int[] a) {
    //PUT YOUR CODE HERE
    for(int i=0; i<a.length; i++) {
        if(a[i]%7 == 0) {
            System.out.println(a[i]);
        }
    }
}
```

5. Write a static method that takes two vectors (1D arrays) of integers, subtracts them, and then returns the results. Also both arrays should be the same size or else return null.

```java
public static int[] subtractVect(int[] vect1, int[] vect2) {
    if(vect1.length != vect2.length)
        return null;
    int[] retVect = new int[vect1.length];
    for(int i=0; i<retVect.length; i++) {
        retVect[i] = vect1[i]-vect2[i];
    }
    return retVect;
}
```
Part 2 – Linked Lists

Building on arrays, we introduce a new way to group together like information. Using nodes which contain data and links, we created a resizable collection of data.

Short Answer

6. What’s one of the greatest advantages of using a linked list over an array?

An array is fixed in size, but a linked list can grow and shrink.

7. Given the linked list below, redraw the links after a new node is inserted after the node that contains 4. Indicate new links by arrows and mark out the broken links. Also assume the new node contains the value 8

![Diagram of linked list]

Programming

8. Write a method that inserts a new value after another a given value. If a node with the given value doesn’t exist put the new node and new value at the end. The method should not return any values and should take in both the value being search for of generic type T and also the value of newly created node of generic type T. Also assume that nodes have both data and link values as seen in given code description.

```java
public void insertAfter(int findValue, int nodeValue)
{
    LinkNode newNode = new LinkNode();
    newNode.data = nodeValue;

    if(head == null)
    {
        head = newNode;
        return;
    }

    LinkNode currNode = head;
    while(currNode != null)
    {
        if(currNode.link == null)
        {
            currNode.link = newNode;
            break;
        }
        currNode = currNode.link;
    }
}
```
currNode.link = newNode;
break;
}
else if(currNode.data == findValue)
{
    newNode.link = currNode.link;
    currNode.link = newNode;
    break;
}
currNode = currNode.link;
}

9. Write a method that removes all instances strictly less than a given value in an integer linked list. This method should not return anything and is given the value which will determine if a node is removed. Also assume that nodes have both data and link values as seen in given code description.

public void removeSmallerThan(int value)
{
    ListNode pTemp = null; //previous temp
    ListNode temp = head;
    while(temp != null)
    {
        if(temp == head && head.data < value)
        {
            head = head.link;
            temp = head;
        }
        else if(temp.data < value)
        {
            pTemp.link = temp.link;
            temp = temp.link;
        }
        else
        {
            pTemp = temp;
            temp = temp.link;
        }
    }
}
Part 3 – Queues

Grouping data on the principle first in first out (FIFO) we learned how add data (enqueue) and remove data (dequeue) in these elegant data structures. It is important to know how the operations enqueue, dequeue, and peek all work using either arrays or linked lists.

10. Given this programming snippet, what will print out at the end? Add is the same as enqueue and remove is the same as dequeue.

```java
Queue<Integer> intQueue = new LinkedList<Integer>();
for(int i=0; i<21; i+=3) {
    intQueue.add(i);
}
while(intQueue.size()>3) {
    intQueue.remove();
}
for(int i : intQueue) {
    System.out.println(i);
}
```

12
15
18
11. Given an initial queue these lines of pseudo-code, fill in how the queue looks after each step.
1. Dequeue 3 times
2. Enqueue 7
3. Enqueue 12
4. Enqueue 23
5. Dequeue 2 times

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 25   | 74   | 84   | 3    | 23   | 85   | 96   |

1

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 3    | 23   | 85   | 96   |      |      |      |

2

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 3    | 23   | 85   | 96   | 7    |      |      |

3

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 3    | 23   | 85   | 96   | 7    | 12   |      |

4

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 3    | 23   | 85   | 96   | 7    | 12   | 23   |

5

| Head | | | | | | | |
|------|------|------|------|------|------|------|
| 85   | 96   | 7    | 12   | 23   |      |      |
12. With the given linked list in this exam, write a method that enqueues a new node with a double value given in the parameter. Make sure to change the links properly to ensure that the data structure remains a queue.

```java
public void enqueue(int data) {
    ListNode newNode = new ListNode(data, null);
    if (head == null) {
        head = newNode;
        tail = head;
        return;
    }
    this.tail.link = newNode;
    this.tail = newNode;
}
```
Part 4 – Stacks

This data structure used the last in first out (LIFO) principle in order to add (push) and remove (pop) data. It’s important to know how the operations push, pop, and peek work using either arrays or linked lists.

Short Answer

13. Given this programming snippet what will print out?

```java
for(double i=0; i<100.0; i+=13.0)
{
    dubStack.push(i);
}
for(int i=0; i<2; i++)
{
    dubStack.pop();
}
dubStack.push(50.0);
while(dubStack.isEmpty()==false)
{
    System.out.println(dubStack.pop());
}
```

50
65
52
39
26
13
0
14. Given this pseudo-code what will the stack look like at each step.

1. Push 23
2. Pop 3 times
3. Push 18
4. Push 22
5. Pop

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

3

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

5

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>74</td>
<td>84</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>
Programming

15. With the given linked list structure write a method for pop for a stack of type int.

```java
public int pop()
{
    if(head != null)
    {
        int retData = head.data;
        head = head.link;
        return retData;
    }
    return 0;
}
```

Integer Linked List for reference

```java
public class IntLinkedList {
    public class ListNode
    {
        int data;
        LinkNode link;
        public ListNode(int aData, ListNode aLink)
        {
            this.data = aData;
            this.link = aLink;
        }
    }

    ListNode head;
    ListNode tail;
}
```