**Question 1:**

This is valid, the LIFO characteristics is Last in, first out. This means the Last item that is inserted in the stack will be the first that will be popped. But when the stack is reversed its no longer following the rule. Because when the 1st and the 10th items swap their positions, then the 10th item is no longer the first one in the list but was the last one in, hence it will become First in, first out.

**Question 2:**

Operation 1 – Invalid because the stack is empty (only stackOne is being filled)

Operation 2- Valid because in the for loop the stack is being filled and return 4.

Operation 3- Valid since the stack is filled it would pop the last item, in this case 4.

Operation 4- Invalid because the stack is empty and there is nothing to peek at.

**Question 3:**

stack.peek() returns the value that is going to be removed when stack.pop() is called. Stack.peek() does not modify the stack size in anyway it just return the value that is last inserted, because this operation’s job is only to return and not remove it.

**Question 4:**

A method will take in a integer value (int n)

Initialize a variable to keep count (int c =0;)

Make a loop to traverse through the linked list to compare with the given integer value (int n) with (c ++;)

When the loop finds a value equal to int n the count stops

The loop breaks and output the count

The count is considered as the index of the first occurrence of int n in the linked list/

**Question 5:**

1. A B / C D – A / +
2. Ans: 4

Push 10 Pop / Push 2 Pop - Push 20 Pop / Push 2 Pop + Push 4

Push 5 Pop 5 Push 30 Pop 10 Push 10 Pop 10 Push + Pop 2

Push / Pop10 Push 10 Pop 30 Push / Pop 20 Pop 2

Push -

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | - |  | / |  |  |  |  |
| / |  | 10 |  | 10 |  | + |  |  |
| 5 |  | 30 |  | 20 |  | 2 |  |  |
| 10 |  | 2 | 2 | 2 | 2 | 2 |  | 4 |

(For some reason on the midterm PDF there is no Question 6)

**Question 7:**

1. This is possible, because when n is 5 (since n >=4) and you simply connect each node with each other, then every single node will have an even degree (like 2).
2. This is not possible (unless one node can make a connection to itself). There will always be on node which will have an even degree no matter how much you modify it. For example, when you take 5 nodes and then try to make all of them represent 3 degrees then one node will always be even.

**Question 8:**

First Graph:

1. Unidirectional and Unweighted Graph
2. Adjacency Matrix:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

1. Yes, because there are no nodes that are isolated or has no all 0s in the matrix.

Second Graph:

1. Directional and Weighted Graph
2. Adjacency Matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nodes | 1 | 2 | 3 | 4 | 5 |
| 1 | inf | 4 | 5 | inf | inf |
| 2 | inf | inf | inf | 6 | inf |
| 3 | inf | 2 | inf | inf | 3 |
| 4 | inf | inf | inf | inf | 4 |
| 5 | inf | inf | inf | inf | inf |

1. No, there many instances like you cannot get to 1 and cannot go anywhere from 5.