**CSCE146 - Practice Exam (Midterm 1)**

CSCE146 F2017 SI | Midterm #1 | For JJ Sheppherd’s Class

**Java Review**

1. What does this Java Code Print out?

|  |  |
| --- | --- |
| public static void main() {  int a = 6;  String s = "";  for (int i = 0; i < a; i++) {  for (int j = 0; j <= i; j++) {  if (j == 0 || j == i) {  s += "\*";  } else {  s += " ";  }  }  System.out.print(s + "\n");  s = "";  }  } | \*  \*\*  \* \*  \* \*  \* \*  \* \* |

1. What error if any will this code segment give? What will it print out if there’s no error?

|  |  |
| --- | --- |
| int[] a = {1,2,3,4,5,6,7,8,9,10};  for (int i = 0; i < a.length / 2; i++) {  System.out.println((a[i + 2] + a[i]));  } | 4  6  8  10  12 |

1. Write a Method that finds the Minimum integer in an array.

public static int findMin(int[] a) {

int check = a[0];

for (int I = 0; i < a.length; i++) {

if (check > a[i]) check = a[i];

}

Return check;

**Data Structures**

**Linked Lists**

Know how to write code to find, delete, and insert Nodes

1. List a few Advantages and Disadvantages of using a Linked List over an Array.

Advantage: Resizable. Disadvantage: Slow Access

1. Draw the Insertion Procedure for adding a node containing 6 after the node containing 5

7

5

1

1

1. Draw the Removal Procedure for the node after 5.

7

5

1

1. Write a Function to find if a value is in the LinkedList using a linear search.

public boolean linearSearch(int toFind) {

Node temp = head;

While (temp != null) {

If (temp.data == toFind) return true;

Temp = temp.next;

}

Return false;

**Queues**

Know how to write code to Enqueue, Dequeue and Peek in a Queue

1. Draw the Queue after each Operation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  | tail |  |  |  |  |
| 5 | 4 | 8 |  |  |  |  |

Enqueue 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  | tail |  |  |  |
| 5 | 4 | 8 | 3 |  |  |  |

Dequeue 3 times

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

Enqueue 6 and 24

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 3 | 6 | 24 |  |  |  |  |

Dequeue 2 times

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |

1. What will the code snippet print out?

|  |  |
| --- | --- |
| Queue<Integer> q = new LinkedQueue<Integer>();  //Assume that this Queue uses enqueue(), dequeue(), and peek()  for (int i = 5; i >= -5; i--) {  q.enqueue(i);  }  for (int i = 3; i < 6; i++) {  System.out.println(q.dequeue());  }  for (int i : q) {  System.out.println(q);  } | 5  4  3  2  1  …  -4  -5 |

**Stacks**

Know how to code Push, Pop, and Peek

1. What will the Code Snippet Print out?

|  |  |
| --- | --- |
| Stack<Integer> s = new LinkedStack<Integer>();  //Assume that this Stack uses pop(), push(), and peek()  for (int i = 5; i >= -5; i--) {  s.enqueue(i);  }  for (int i = 3; i < 6; i++) {  System.out.println(s.dequeue());  }  for (int i : s) {  System.out.println(s);  } | -5  -4  -3  -2  …  4  5 |

1. Draw the Stack after each Operation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  | tail |  |  |  |  |
| 5 | 4 | 8 |  |  |  |  |

Push 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 3 | 5 | 4 | 8 |  |  |  |

Pop 3 times

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |

Push 6 and 24

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 24 | 6 | 8 |  |  |  |  |

Pop 2 times

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |

**Recursion**

1. What data Structure can be used to illustrate Recursion?

Stacks

1. What does this code do?

|  |  |
| --- | --- |
| public static int f(int a) {  if (a <= 1) return 1;  return f(a - 1) + a;  } | Returns the Triangular Number of f(a) {a + (a-1) + … + 2 + 1} |

**Searching and Sorting**

Array: {45,23,12,79,36,42,10}

1. Perform Mergesort on the Given Array

{45, 23, 12, 79} {36, 42, 10}

{45, 23} {12, 79} {36, 42} {10}

{45} {23} {12} {79} {36} {42} {10}

{23, 45} {12, 79} {36, 42} {10}

{12, 23, 45, 79} {10, 36, 42}

{10, 12, 23, 36, 42, 45, 79}

1. Perform Quicksort on the Given Array
2. Perform a Binary Search for 45 for the given array (After it has been sorted)

{10, 12, 23, 36, 42, 45, 79}

{10, 12, 23, **36**, 42, 45, 79}

{10, 12, 23, 36} {42, **45**, 79}

**Asymptotics**

Know how to sort them

Know what Complexity the Algorithms in class have