**Name:**\_\_\_\_Ulvi Bajarani\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SID **#:**\_\_\_20539914\_\_\_\_\_\_\_

**Matching** (6 points each)

*For each term in the left column, write the letter for the description that best matches the term from the right column.*

\_\_e\_\_ 1. Stack

\_\_f\_\_ 2. Hash function

\_\_b\_\_ 3. Linked list

\_\_c\_\_ 4. Queue

\_\_a\_\_ 5. Load factor

\_\_d\_\_ 6. Priority queue

1. The number of records in the hash table divided by the size of the hash table.
2. Linear collection of self-referential class objects called nodes connected by reference links.
3. First In First Out data structure.
4. Customers or jobs with higher priority are pushed to the front of the data structure.
5. Last In First Out data structure.
6. Takes a search key and produces the integer index of an element in the hash table.

**True/False** (6 points each)

*Indicate whether the sentence or statement is true or false.*

\_\_F\_7. f(n) = 4n + 2n2 + 5  O(n).

\_\_T\_8. A sequential search of a list assumes that the list is in ascending order.

\_\_F\_9. Binary search can be performed on both sorted and unsorted lists

**Multiple Choice** (6 points each)

*Identify the letter of the choice that best completes the statement or answers the question.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] |
| list | B | D | F | H | L | N | P | R | U |

\_\_**b**\_10. Using binary search algorithm, how many key comparisons would have to be made on the list above to find the letter P?

|  |  |  |  |
| --- | --- | --- | --- |
| a. | 7 | c. | 6 |
| **b.** | 2 | d. | 5 |

**Short-Answer questions** (10 points each)

lastNode

*Answer the following questions in the space provided.*

firstNode

newNode

1. Assume the above setting of a linked list. Write the Java code to insert the new node referenced by *newNode* into the list referenced by *firstNode* at the front of the list.

***// Assuming that for exercises 11-14 the definition of nodes, firstNode and lastNode are already defined:***

**if** (isEmpty())

{

firstNode = lastNode = new newNode<E>(insertItem);

}

**else**

{

firstNode = new newNode<E>(insertItem, **firstNode**); // Setting first node as next node first, then inserting the data.

}

lastNode

firstNode

newNode

1. Assume the above setting of a linked list. Write the Java code to insert the new node referenced by *newNode* into the list referenced by *firstNod*e between nodes whose values are 65 and 34.

lastNode

firstNode

newNode<E> current = firstNode;

while (current.nextNode != null)

{

if (current.data == 65 && current.nextNode.data == 34) // if the order is 65 and 34.

{

firstNode = new newNode<E>(insertItem, **current.nextNode**); // Making *current* as the firstNode of the place where it is.

break;

}

}

1. Assume the above setting of a linked list. Write the Java code to delete the node whose value is 34 from the list.

lastNode

firstNode

. . .

newNode<E> current = firstNode;

while (current.nextNode != null)

{

if (current.data == 34 && current.nextNode != lastNode)

{

firstNode=current.nextNode; // Delete from the front of the node in the middle of the list.

break;

}

else if (current.data == 34 && current.nextNode == lastNode)

{

firstNode=lastNode=null; // Delete from the front of the node in the end of the list.

break;

}

else

{

System.out.println(“Data wasn’t found.”);

}

}

1. Assume the above setting of a linked list with *n* number of nodes. Write the Java code to count how many nodes are in the list.

newNode<E> current = firstNode;

**int** count = 0;

**while** ( current.nextNode != **null**)

{

++count;

}

System.out.printf(“the number of nodes are %d\n”, **count**);