وقت المراجعة

LIVE

الجمعة والسبت

7+8 Nov 2025

07:30 PM

09:30

OL Academy

مدرس المقرر

أحمد كريم

ITCS214

Data Structures

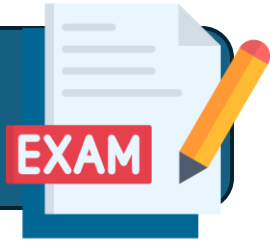
Exam1

Midterm Exam Revision

سجّل الآن
عبر موقعنا

www.olearninga.com olearninga 66939059

QUESTION #1:



Output questions

Array-Based Lists

Show the output of the following:

```
public static void main(String[] args) {
ArrayList<Integer> list1 = new ArrayList<>();
ArrayList<Integer> list2 = new ArrayList<>();
list1.add(5);  list1.add(4); list1.add(11);
list1.add(7);  list1.add(6);  list1.add(3,10);
list1.set(2, list1.remove(list1.size()-2));
list1.add(list1.indexOf(4));
System.out.print("list1: ");
for (int i = 0; i < list1.size(); i++)
System.out.print(list1.get(i) + " ");
for (int i = 0; i < list1.size(); i++) {
   int num = list1.get(i);
   if (num % 2 == 0)
      list2.add(num * num);
   else
      list2.add(num);
}
System.out.print("\nlist2: ");
for (int i = 0; i < list2.size(); i++) {
   System.out.print(list2.get(i) + " ");
 }
}
```

Output

B Single Linked List

Assume you are given the following **CompareList** method, which is included in class **SingleTest.** Show the output of the main method after calling **CompareList** method.

```
public class SingleTest <E> {
public static<E> boolean CompareList (SingleLinkedList<E> list1, E
item1)
{
    if (list1.size()==0)
        return false;
   E maxnum = list1.get(0);
   int x = 1;
   while (x < list1.size())
   {
       E item2 =list1.get(x);
       if(((Comparable)item2).compareTo((Comparable) maxnum)>= 0 )
       maxnum = item2 ;
      x++;
   }
   if (((Comparable)item1).compareTo((Comparable) maxnum) > 0 )
       list1.add(list1.size(),item1);
   else
   {
       int indexMax =list1.indexOf(maxnum);
       list1.remove(indexMax);
    }
 return true;
}
```

```
public static void main(String[] args)
{
SingleLinkedList <Integer> SLL = new
SingleLinkedList<>();
SLL.add(14);
SLL.add(10);
SLL.add(25);
SLL.add(5);
SLL.add(9);
CompareList( SLL, 20);
System.out.print("\n List A: ");
for(int i=0; i< SLL.size(); i++)
System.out.print(SLL.get(i)+" ");
CompareList( SLL, 30);
System.out.print("\n List B: ");
for(int i=0; i< SLL.size(); i++)
System.out.print(SLL.get(i)+" "); }}
```

Output

C Double Linked List

Show the output of the following;

```
KWLinkedList<Integer> mylist = new KWLinkedList<>();
mylist.addFirst(10);
mylist.addFirst(12);
mylist.addLast(5);
mylist.add(2, 7);
mylist.add(4);
ListIterator<Integer> iter1 = mylist.listIterator(1);
while (iter1.hasNext())
    System.out.print(iter1.next()+" ");
System.out.println();
iter1.add(9);
iter1.add(3);
iter1.previous();
while (iter1.hasPrevious())
    System.out.print(iter1.previous()+" ");
System.out.println();
iter1.remove();
while (iter1.hasNext())
    System.out.print(iter1.next()+" ");
System.out.println();

iter1.set(15);
System.out.print(iter1.hasNext()+" ");
System.out.print(iter1.previousIndex() +" ");
iter1 = mylist.listIterator(3);
while (iter1.hasNext())
    System.out.print(iter1.next()+" ");
```

Output

QUESTION #2:

Single Linked List

Write a method called **copySpecial** to be included in the SingleLinkedList class. The method has two parameters, **S** of type SingleLinkedList, which is initially empty, the method copies nodes from "this" list to S list if the value of the node is greater than item. You are not allowed to call any of the SingleLinkedList class methods (you can create nodes). The method head is:

public void copySpecial (SinleLinkedList<E> S, E item)

Before method call: item: 7	After method call:
head this: [4 20 25 26 -3 10] S:	head this: [4 20 25 26 -3 10] head S: [20 25 26 10]

QUESTION #3:  Double Linked List

Write a method called **containsRepeatedItem** to be included in an application class called KWLinkedListApplication that accepts **list1** of type KWLinkedList<Integer>. The method returns true if list1 contains a repeated element; it returns false, otherwise.

The method head is
public static boolean containsRepeatedItem (KWLinkedList <Integer> list1)

Note: You must use the ListIterator(s) and its methods only.

Example:
If list1: [13 14 13 15 16 17] then the method returns **true** because 13 is repeated.
If list1: [13 14 19 16 17] then the method returns **false** as there is no repeated item.

QUESTION #4:  Stacks

Evaluate the following postfix expression using stacks showing the stack operation one by one.

45 3 3 / * 5 + 10 / -8 -

b) Write a method called **copyHalf** to be included in an application class called StackApplication. The method has three parameters **S1, S2** and **S3** of type ArrayStack. The method copies the first half of the elements to **S2** and copies the second half of the elements to **S3 (as shown in the example given below),** if S1 has **even number** of elements only. Initially, S1 was not empty but S2 and S3 are empty.
Assume that the class ArrayStack is available for use. Use common stack operations only such as push, pop, peek, isEmpty, and copy constructor. Do not use any other data structure in your method.

The method header is given below:
public static<E> void copyHalf(ArrayStack<E> S1, ArrayStack<E> S2, ArrayStack<E> S3)

Example1:

Before method call:
top
S1: 23 29 22 17 16 7 11 9
S2: empty
S3: empty

After method call:
S1: empty
top
S2: 17 22 29 23
S3: 9 11 7 16

Example2:

Before method call:
top
S1: 23 29 22 17 16 7 11 9 12
S2: empty
S3: empty

After method call:
top
S1: 23 29 22 17 16 7 11 9 12
S2: empty
S3: empty