OL Academy

Second Semester, 2024-2025
ITCS214 (Data Structures)

Midterm Exam Revision

Array Based Lists

Question (1) : Assume that list1 is an object of class type ArrayList<Integer> of java (similar to KWArrayList class) and it has the following values:

4 6 3 2 1

The instruction:

list1.set(1,get(indeoxOf(2));

Will change the list to:

(A) 1 6 8 2 1

(B) 3 6 3 2 1

(C) 3 4 6 2

(D) 4 2 3 2 1

<u>Question (2):</u> Assume that list1 is an object of class type KWArrayList<Integer> class and it has the following values:

5 9 1 7 1

<u>The instruction</u>

list1.set(2,list1.lastIndexOf(0));

Will change the list to:

Will change the list to:

(A) 5 4 5 7 1
(B) 5 9 5 7 1
(C) 5 9 -1 7 1
(D) 5 -1 1 7 1

Question (3): Find the output of the following code

```
KWArrayList <String> namesList = new KWArrayList<>();
namesList.add("Mohammed");
namesList.add("Noor");
namesList.add("Jood");
namesList.add("Isa");
namesList.add(2,"Mariam");
namesList.set(1,"Sara");
int i= namesList.size()-1;
while (!namesList.isEmpty() && i !=0)
        System.out.println(namesList.get(i--));
```

Output

Question (4): Consider the generic class**KWArrayList**(as discussed in the lectures) having following data fields (private):
private static final int INITIAL_CAPACITY = 10;// The default initial capacity
private E[] theData; // The underlying data array
private int size ; // The current size
private int capacity ; // The current capacity
This class has the following methods:

Method	**Behavior**
public KWArrayList ( )	Default Constructor with capacity = INITIAL_CAPACITY
public KWArrayList (int cap)	Constructor with capacity = cap
public int size()	Returns current size
public boolean contains(E obj)	Checks whether the given object obj is present in the list. If it is there then it returns true else it returns false.
public void clear()	Removes all the elements of the array list and make it empty
public boolean isEmpty()	Checks whether list is empty or not
public boolean add (E anEntry)	Adds object anEntry at the end of the list and returns true.
public void add (int index, E anEntry)	Adds object anEntry in the list at the location given by index
public E get (int index)	Returns the element of the list at position given by index
public E set (int index, E newValue)	Updates the element at position index by newValue and returns the old value
public E remove (int index)	Removes the element at position index and returns the element being removed
public boolean remove (E obj)	Removes the first occurrence of the object obj from the list, if present and returns true, else returns false.
private void reallocate ()	Private method to expand the array by allocating a new array of double the previous capacity. Called if the list becomes full
public int indexOf(E obj)	Returns the index of the first occurrence of the specified element obj in this list, or -1 if this list does not contain the element.
public String toString()	Returns the String equivalent of the list object

Answer following questions:

(A) Write a method called **removeMid** to be included in class **KWArrayList** that don't have any parameter. The method will remove the mid item if it is repeated in the list and return true. If the list is empty or the item not repeated don't do anything and return false.

You should be sure first that the number of elements in the list is odd, otherwise the method return false.

Method heading:

public boolean removeMid()

Do not call any method of the class **KWArrayList** in your method.
Example:

list1:

index	0	1	2	**3**	4	5	6
data	10	15	25	**20**	25	20	17

The method will remove the 20 in position 3 and return **true**.

(B) Write a method called **checkItem** in an application class called **ListApplication** that has first parameter **list1** of type **KWArrayList** and second parameter **item** of type int. If all the elements in the list before **item** are smaller than it, and all the elements in the list after **item** are greater than it, then the method will return true. In all other cases, the method will return false. Assume that **list1** has at least 3 elements.

Example 1: **item = 10**
list1: 5 6 3 9 2 **10** 17 12 30
In this case the method will return **true**.

Example 2: **item = 10**
list1: 5 6 15 18 2 **10** 7 12 30
In this case the method will return **false**.

(C) Write a method called **ChangeMinAndMax** to be included in an application class called **ArrayListApplication** that accepts one parameter **list1** of type **KWArrayList**. The method finds the maximum element and the minimum element of the list. Then place the minimum element **at the beginning of list1** and the maximum element **at the end of list1**. If the list is empty or has only one element, it returns false, otherwise, it returns true.

public static<E> boolean ChangeMinAndMax (KWArrayList <E> list1)

Note: Write this method by calling methods of the class ***KWArrayList****.*

Example:
Before method call: list1: [30 35 15 **5** 20 25 **40** 10]

After method call: list1: [**5** 30 35 15 20 25 10 **40**] and returns true.

Single Linked List

<u>Question (5):</u>
What is the output of the following code?

```
SingleLinkedList<Integer> list = new
SingleLinkedList<>(); list.add(5);
list.add(7);
list.add(12);
list.add(15);
list.add(1, 100);
for(int i=0; i<list.size(); i++)
     System.out.print(list.get(i) + “ ”);
 list.set(2, 50);
 list.add(10)
 list.remove(2);
 System.out.println();
 for(int i = 0; i < list.size(); i++)
     System.out.print(list.get(i) + “ ”);
```

Output:

Question (6):

Write a method called **copyAlternate** to be included in the class **SingleLinkedList** that accepts a parameter **list1** of type **SingleLinkedList**. If "this" list is empty, it will return false, otherwise, it will copy alternative nodes from "this" list to **list1** and returns true. Assume that **list1** is initially empty. Do not call any method of class **SingleLinkedList**.

Method heading:

public boolean copyAlternate(SingleLinkedList<E> list1)

Note: Copy here means creating and adding new nodes in **list1**. You can use constructor of class **Node** to create new nodes.

Example :

Before method call:	"this" list:	7 8 10 9 6
	list1:	(empty)
After method call:	"this" list:	7 8 10 9 6
	list1:	7 10 6

Double Linked List

Question (7): What Would be the output of the following code?

```
KWLinkedList<Integer> list = new KWLinkedList <Integer> ();
for (int i=0; i<4; i++)
        list.add (i);
ListIterator <Integer> iter = list.listIterator (2);
while (iter.hasNext()){
        int x = iter.next ();
        if (x % 2 == 1)
                iter.set (x+1);
}
while (iter.hasPrevious ())
        System.out.print (iter.previous () + "     " );
```

Output:

Question (8):

Write a method called **reverseNodes** to be included in the **KWLinkedList** class which reverses the nodes of a doubly linked list, Write the method by swapping the data fields of the nodes, if the list is empty or contains only one node display this message (List is Empty or has one node only), Assume that the list contains even number of elements.

Write your method by using listIterator.

Method heading: public void **reverseNodes ()**

Do not call any method of class KWLinkedList in your method.

```
public  void reverseNodes ()
{
```

Question (9): Given the following program segment, show the output in the box provided:

```
KWLinkedList<Integer> list1 = new KWLinkedList<Integer>();
list1.addLast(10);
list1.addLast(5);
list1.addLast (15);
list1.addLast (20);
list1.addLast (9);
list1.addLast (3);
ListIterator<Integer>  iter = list1.listIterator();
int a, b;
while(iter.hasNext())
{
     a= iter.next();
     b= iter.next();
     if (a> b)
     iter.set(a + b);
}
while(iter.hasPrevious())
      System.out.println(iter.previous() + "  ");
```

Question (10):

Write a method called **addBefore** to be included in class **KWLinkedList**(a class for doubly linked list).The method accepts two parameters **item1** and **item2** of type **E.** If **item1** exists in the list then the method will insert **item2** before the first occurrence of **item1** and will return true, else the method will not insert any item and will return false.

Do not call any method of class **KWLinkedList,** also do not use any iterator in your method. Consider all cases, such as the list is empty, **item1** is in the first node, **item1** is anywhere in the list and **item1** does not exists in the list.

Example:

item1: 10 item2 : 15

Iist (before method call): 7 12 **10** 20 14 **10** 5

Iist (after method call) : 7 12 **15** **10** 20 14 **10** 5

```
public boolean addBefore (E item1,E item2)
{
```

Stacks

Question (11):

(a) What is the output of the following program?

```
public class StackTest
{
        public satic viod main(String[] args)
        {
                ArrayStack<Integer> st1=new ArrayStack<Integer>();
                ArrayStack<Integer> st2=new ArrayStack <Integer>();
                for(int i = 1; i <= 6; i++)
                       st1.push(i);
                 while(!st1.isEmpty()) {
                      int item = st1.pop();
                      if(item % 2 == 0)
                             st2.push(item);
                 }
                 while(!st2.isEmpty())
                        st1.push(st2.pop());
                 while(!st1.isEmpty())
                       System.out.print(st1.pop() + "  ");
        }
}
```

Output:

(b) Evaluation of the following postfix expression using stacks is

12 24 6 / + 5 3 - 10 * -

<u>Question (12):</u> Write a method called **deleteAboveKey** in a class called **`StackEx`** that accepts an object **st1** of type **ArrayStack** as the first parameter and **key** of type **E** as the second parameter. The method will delete all elements from the stack st1 which are greater than or equal to the key, *<u>starting from the first occurrence of the key</u>*. All the remaining elements of **st1** should be in the original relative order. If stack **st1** is empty or key is not found in st1, then do not delete any element from **st1** and return false, otherwise return true.

Example: key=9

Stack **st1** before method call

14	2	15	**9**	32	5	8	40
Top							

Stack **st1** after method call

14	2	15	**9**	5	8

Assume that class ArrayStack is available for use. Use common stack operations only such as push, pop, peek, isEmpty and copy constructor.

Public class `StackEx{`