Data Structures CSC 212	Midterm Exam - Fall 2020
Date: 13/10/2020	Duration: 90 minutes

Guidelines

- No calculators or any other electronic devices are allowed in this exam.

Student ID:

Section: Name:

Instructor:

1	2	3	4	Total

Question 1 ..... 15 points

Clearly mark *one* answer for each of the following:

1. For $f(n) = 2n^{\frac{7}{2}} + 6n^2 - n^3$, which of the following is *true*? $f(n)$ is:
   Ⓐ $O(n^{\frac{7}{2}})$ Ⓑ $O(n^3)$ Ⓒ $O(n^2)$ Ⓓ A and B Ⓔ None
2. For $f(n) = 14n + 2n\log n - 6n^{\frac{1}{2}}$, which is the *most appropriate*? $f(n)$ is:
   Ⓐ $O(n)$ Ⓑ $O(n^{1/2})$ Ⓒ $O(n^2)$ Ⓓ $O(n\log n)$ Ⓔ None
3. For $f(n) = \log(n+1)^2$, which is the *most appropriate*? $f(n)$ is:
   Ⓐ $O(\log n)$ Ⓑ $O((\log n)^2)$ Ⓒ $O(\log(n^2))$ Ⓓ $O(n^2)$ Ⓔ None
4. Suppose you had an algorithm $\mathcal{A}$. When it runs on a list l of size n, $\mathcal{A}$ calls algorithm $\mathcal{B}$ on each of the first three iterations. On the remaining iterations, it calls algorithm $\mathcal{C}$. Suppose $\mathcal{B}$ is $O(n)$ and $\mathcal{C}$ is $O(1)$. Which running time *best describes* algorithm $\mathcal{A}$:
   Ⓐ $O(n)$ Ⓑ $O(n^2)$ Ⓒ $O(2^n)$ Ⓓ $O(1)$ Ⓔ None
5. Suppose that an algorithm $\mathcal{A}$ iterates over a list l of size n. For each element in the first half of l $\mathcal{A}$ executes some $O(\log n)$ operation. For elements in the second half of l, $\mathcal{A}$ executes some $O(1)$ operation. Which running time *best describes* algorithm $\mathcal{A}$?
   Ⓐ $O(n)$ Ⓑ $O(n\log n)$ Ⓒ $O(n+\log n)$ Ⓓ $O(n^2\log n)$ Ⓔ None
6. For two functions $f(n)$ and $g(n)$, suppose $f(n)$ is $O(g(n))$. What is the best big-O for $f(n)+g(n)$?
   Ⓐ $O(g(n))$ Ⓑ $O(2f(n))$ Ⓒ $O(f(n)\cdot g(n))$ Ⓓ $O(f(n)/g(n))$ Ⓔ None
7. Suppose $f(n)$ is $O(g_1(n))$ and $h(n)$ is $O(g_2(n))$. What is the best big-O of $f(n)\cdot h(n)$?
   Ⓐ $O(g_1(n)+g_2(n))$ Ⓑ $O(g_1(n))$ Ⓒ $O(g_1(n)\cdot g_2(n))$ Ⓓ $O(g_2(n))$ Ⓔ None

Scanned with CamScanner

Question 2 ..... 25 points

In a real estate management application, information about properties (e.g., homes, farms, and land) and their owners are stored in the two classes `Property` and `Owner` shown below. Notice that a property may have multiple owners, and an owner may participate in owning several properties.

```
public class Property {
  public String id;
  public String propertyName;
  public double value;
  public List<Owner> owners; // Owners of this property
  public Property(String id, String propertyName, List<Owner> owners) {
    this.id= id;
    this.propertyName= propertyName;
    this.owners= owners;}
}
```

```
public class Owner {
  public String firstName;
  public String lastName;
  public Owner(String firstName, String lastName) {
    this.firstName= firstName;
    this.lastName= lastName;}
}
```

Write the method `ownersOf` that takes as input a list of properties and a particular property's ID, and checks if that property is included in the list. The method should return a list containing all the owners of this property if it exists; otherwise it should return an empty list. The method signature is: `static List<Owner> ownersOf(List<Property> properties, String id)`.

1. Line 1:
   - (A) `if (!properties.empty()){`
   - (B) `while (!properties.last()){`
   - (C) `properties.findFirst();`
   - (D) `if (properties.head != null){`
   - (E) None
2. Line 2:
   - (A) `if (!properties.empty()){`
   - (B) `if (properties.retrieve().id.equals(id))`
   - (C) `properties.current = head;`
   - (D) `properties.findFirst();`
   - (E) None
3. Line 3:
   - (A) `return properties.retrieve().owners;`
   - (B) `while (properties.last()){`
   - (C) `while (!properties.last()){`
   - (D) `while (properties.current.next != null){`
   - (E) None
4. Line 4:
   - (A) `properties.findNext(); }`
   - (B) `if (properties.retrieve().id.equals(id))`
   - (C) `if (properties.retrieve().id == id)`
   - (D) `if (properties.data.id.equals(id))`
   - (E) None
5. Line 5:
   - (A) `if (properties.retrieve().id.equals(id))`
   - (B) `return properties.data.owners;`
   - (C) `return properties.owners;`
   - (D) `return properties.retrieve().owners;`
   - (E) None
6. Line 6:
   - (A) `properties.findNext(); }`
   - (B) `properties.current = current.next; }`

Scanned with CamScanner

Ⓒ `return properties.retrieve().owners; }`

Ⓓ `properties.findFirst(); }`

Ⓔ None

7. Line 7:

Ⓐ `if (!properties.retrieve().owners.empty())`

Ⓑ `if (properties.retrieve().id == id)`

Ⓒ `if (properties.data.id.equals(id))`

Ⓓ `if (properties.retrieve().id.equals(id))`

Ⓔ None

8. Line 8:

Ⓐ `properties.retrieve().owners.remove(); }`

Ⓑ `return properties.retrieve().owners; }`

Ⓒ `return properties.owners; }`

Ⓓ `return properties.data.owners; }`

Ⓔ None

9. Line 9:

Ⓐ `return new LinkedList<Property>();`

Ⓑ `return new LinkedList<Owner>();`

Ⓒ `return properties;`

Ⓓ `return null;`

Ⓔ None

Question 3 ..... 25 points

Clearly mark *one* answer for each of the following:

1. Which of the following is true about linked implementation of queue?

Ⓐ In enqueue operation, if new nodes are inserted at the beginning, then in serve operation, nodes must be removed from the end. Ⓑ In enqueue operation, if new nodes are inserted at the end, then in serve operation, nodes must be removed from the beginning. Ⓒ Both of the above. Ⓓ None of the above.

2. What does the method `f` below do ?

```
public static <T> void f(Queue<T> q) {
  if (q.length() > 0) {
    T e = q.serve();
    f(q);
    q.enqueue(e);
  }
}
```

Ⓐ Leaves `q` unchanged. Ⓑ Empties `q`. Ⓒ Deletes the first element of `q` and inserts it at the end keeping the other elements in the same order. Ⓓ Reverses `q`. Ⓔ None

3. Suppose we have a circular array implementation of the queue, with ten items in the queue stored at `data[2]` through `data[11]`. The capacity (that is `maxSize`) is 42. Where does the enqueue method place the new entry in the array?

Ⓐ data[1] Ⓑ data[11] Ⓒ data[12] Ⓓ data[0] Ⓔ None

4. The function `g` below is member of `ArrayQueue`. What does it do?

```
public T g() {
  if(size == 0)
    return null;
  else {
    T e = data[head];
```

Scanned with CamScanner

```
    return e;
  }
}
```

(A) Return the front element. (B) Enqueue. (C) Serve. (D) Return the last element.
(E) None.

5. In the linked implementation of a queue, which of the pointers `head` and `tail` will change during an enqueue into a **non-empty** queue?

(A) Only `head`. (B) Only `tail`. (C) Both `head` and `tail`. (D) Depends on the size of the queue.
(E) None.

6. What is the content of `q` at the end of the following code:

```
Queue<Integer> q = new LinkedQueue<Integer>()
q.enqueue(5);
q.serve();
q.enqueue(3);
q.enqueue(2);
q.enqueue(4);
q.serve();
q.serve();
q.enqueue(2);
```

~~5~~
~~3~~ ~~2~~ 4
4 2

(A) 5, 3, 2 (B) 5, 3, 2, 4 (C) 4, 2, 3 (D) 2, 4, 2, 3 (E) None

Question 4 ..... 35 points

We want to write a linked implementation of the ADT `UQueue` which is a linear structure that stores elements without repetition and allows to serve from both ends.

```
public interface UQueue<T> {
  int length();
  boolean full();
  // Insert e at the end if it does not already exist and return true, otherwise return false.
  boolean enqueue(T e);
  // Remove and return the first element (the oldest) same
  T serveFirst();
  // Remove and return the last element (the newest)
  T servLast();
}
```

Complete the class `LinkedUQueue` below.

```
class Node<T> {
  public T data;
  public Node<T> next, prev;
  public Node(T data) {
    this.data = data;
    prev = next = null;
  }
}
public class LinkedUQueue<T> implements UQueue<T> {
  private Node<T> head, tail;
  private int size;
  public LinkedUQueue() {
    tail = head = null;
    size = 0;
  }
  public int length() {
    return size;
```

Scanned with CamScanner

```
}
public boolean full() {
  return false;
}
```

1. Method enqueue.

```
1  public boolean enqueue(T e) {
2    if (size == 0) {
3      ...
4    } else {
5      Node<T> p = ...
6      while (...)
7        ...
8      if (...)
9        ...
10     ...
11     ...
12     ...
13   }
14   ...
15   ...
16 }
```

- Line 3:
  - (A) `tail = new Node<T>(e);`
  - (B) `tail = head = new Node<T>(e);`
  - (C) `tail = head = null;`
  - (D) `head = new Node<T>(e);`
  - (E) None
- Line 5:
  - (A) `Node<T> p = head;`
  - (B) `Node<T> p = null;`
  - (C) `Node<T> p = tail.prev;`
  - (D) `Node<T> p = head.next;`
  - (E) None
- Line 6:
  - (A) `while (p.next != null && !e.equals(p.data))`
  - (B) `while (p != tail && !e.equals(p.data))`
  - (C) `while (e.equals(p.data))`
  - (D) `while (p != null && !e.equals(p.data))`
  - (E) None
- Line 7:
  - (A) `p = head.next;`
  - (B) `p.next = p;`
  - (C) `p = p.next;`
  - (D) `p = p.prev.next;`
  - (E) None
- Line 8:
  - (A) `if (p == e)`
  - (B) `if (p.equals(e))`
  - (C) `if (p == head)`
  - (D) `if (p != null)`
  - (E) None
- Line 9:
  - (A) `return p != tail;`
  - (B) `return false;`
  - (C) `return true;`
  - (D) `return p != null;`
  - (E) None
- Line 10:
  - (A) `tail.next = new Node<T>(e);`
  - (B) `tail.prev = new Node<T>(e);`
  - (C) `tail = new Node<T>(e);`
  - (D) `head.next = new Node<T>(e);`
  - (E) None
- Line 11:
  - (A) `tail.prev.next = tail;`
  - (B) `head.prev.next = tail;`
  - (C) `tail.next = tail;`
  - (D) `tail.next.prev = tail;`
  - (E) None
- Line 12:
  - (A) `head = head.prev;`
  - (B) `head.next.prev = tail;`

Scanned with CamScanner

(C) `tail = tail.next;`

(D) `tail = tail.prev;`

(E) None

- Line 14:

(A) `if (head.next != null)size++;`

(B) `size++;`

(C) `size--;`

(D) `if (tail.prev != null)size++;`

(E) None

- Line 15:

(A) `return head.next != null;`

(B) `return e != null;`

(C) `return true;`

(D) `return e;`

(E) None

2. Method `serveFist`.

```
1  public T serveFirst() {
2    T e = ...
3    ...
4    if (...)
5      ...
6    else
7      ...
8    ...
9    ...
10 }
```

- Line 2:

(A) `T e = new Node<T>(head.data);`

(B) `T e = tail.data;`

(C) `T e = head;`

(D) `T e = head.data;`

(E) None

- Line 3:

(A) `tail = tail.next;`

(B) `head = head.prev;`

(C) `head = head.next;`

(D) `tail = tail.prev;`

(E) None

- Line 4:

(A) `if (tail == head)`

(B) `if (tail == null)`

(C) `if (head == null)`

(D) `if (size == 0)`

(E) None

- Line 5:

(A) `head.prev = null;`

(B) `head = null;`

(C) `tail = null;`

(D) `head = tail;`

(E) None

- Line 7:

(A) `head.next = null;`

(B) `tail.next = null;`

(C) `tail.prev = null;`

(D) `head.prev = null;`

(E) None

- Line 8:

(A) `size = 0;`

(B) `size--;`

(C) `size++;`

(D) `if (head !=null)size--;`

(E) None

- Line 9:

(A) `return head.data;`

(B) `return tail.data;`

(C) `return e.data;`

(D) `return e;`

(E) None

Scanned with CamScanner

s. Method serveLast.

```
public T serveLast() {
  T e = ...
  ...
  if (...)
    ...
  else
    ...
  ...
  ...
}
```

- Line 2:
  - (A) `T e = head.data;`
  - (B) `T e = new Node<T>(head.data);`
  - (C) `T e = tail.data;`
  - (D) `T e = head;`
  - (E) None
- Line 3:
  - (A) `head = head.prev;`
  - (B) `tail = tail.next;`
  - (C) `head = head.next;`
  - (D) `tail = tail.prev;`
  - (E) None
- Line 4:
  - (A) `if (tail == head)`
  - (B) `if (size == 0)`
  - (C) `if (head == null)`
  - (D) `if (tail == null)`
  - (E) None
- Line 5:
  - (A) `head = tail;`
  - (B) `head.prev = null;`
  - (C) `head = null;`
  - (D) `tail = null;`
  - (E) None
- Line 7:
  - (A) `tail.next = null;`
  - (B) `tail.prev = null;`
  - (C) `head.prev = null;`
  - (D) `head.next = null;`
  - (E) None
- Line 8:
  - (A) `size++;`
  - (B) `size--;`
  - (C) `if (head !=null)size--;`
  - (D) `size = 0;`
  - (E) None
- Line 9:
  - (A) `return head.data;`
  - (B) `return e.data;`
  - (C) `return e;`
  - (D) `return tail.data;`
  - (E) None

Scanned with CamScanner

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Quiz 3 - Fall 2018**
Date: 25/10/2018	Duration: 25 minutes

Question 1 ..... 24 ..... 30 points

Write the method **public static <T> void moveAfter(Stack<T> st1, Stack<T> st2, int i)** (user of the Stack ADT), that takes as input two stacks st_1, st_2 and an index i. It moves the elements of stack st_2 after the element at position i in stack st_1. Assume that i is within the range of stack st_1, and that the top element has an index 0. The stack st_2 must **not change** after the call.

Example 1. *If st_1 (top to bottom): 5, 2, 4, 1 and st_2 (top to bottom): 8, 9. After calling* `moveAfter(st1, st2, 1)`, *st_1 will be (top to bottom): 5, 2, 8, 9, 4, 1.*

1. Line 1:
   - (A) `Node<T> p = st1.top, q = st2.top; int c = 0;`
   - (B) `Queue<T> x = new ArrayQueue<T>(100);`
   - (C) `Stack<T> x = new LinkedStack<T>();`
   - (D) `T[] x = new T[100];`
   - (E) None
2. Line 2:
   - (A) `while (!st1.empty)`
   - (B) `for(int j = 0; j <= i; j++)`
   - (C) `while(p != null){`
   - (D) `for(int j = 0; j < i; j++)`
   - (E) None
3. Line 3:
   - (A) `x[j] = st1.pop();`
   - (B) `x.push(st1.pop());`
   - (C) `if(c == i)break;`
   - (D) `x.enqueue(st1.pop());`
   - (E) None
4. Line 4:
   - (A) `Stack<T> t = st2;`
   - (B) `Stack<T> t = new LinkedStack<T>();`
   - (C) `Queue<T> t = new LinkedQueue<T>();`
   - (D) `Queue<T> t = st2;`
   - (E) None
5. Line 5:
   - (A) `while(!st1.empty())`
   - (B) `while(st2.empty())`
   - (C) `while(st2.top != null)`
   - (D) `while(!st2.empty())`
   - (E) None
6. Line 6:
   - (A) `t.push(st2.top.data);`
   - (B) `t.push(st2.pop());`
   - (C) `t.enqueue(st2.pop());`
   - (D) `st2.push(t.pop());`
   - (E) None

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212 Date: 00/00/0000	**Quiz 2** Duration: 20 minutes

Student ID:	Name:
Section:	Instructor:

1	2	Total/30

Question 1 ..... 30 points

Write the method **public static <T> void** `cls(List<T>l, int k)` that performs `k` circular left shifts on the list.

Example 1. *If* $l : A \to B \to C \to D \to E$, *then after calling* `l.cls(1)`, $l : B \to C \to D \to E \to A$. *If instead we call* `l.cls(2)`, $l : C \to D \to E \to A \to B$.

1. Line 1:
   - (A) **for** `(int i = 0; i < k; i++){`
   - (B) **while** `(!l.last()){`
   - (C) **while** `(!l.empty()){`
   - (D) **while** `(!l.full()){`
   - (E) None
2. Line 2:
   - (A) `l.findFirst();`
   - (B) `l.findNext();`
   - (C) `l.remove();`
   - (D) `l.insert(l.retrieve());`
   - (E) None
3. Line 3:
   - (A) `l.findFirst();`
   - (B) `T e = l.serve();`
   - (C) `l.findNext();`
   - (D) `T e = l.retrieve();`
   - (E) None
4. Line 4:
   - (A) `l.findNext();`
   - (B) `l.update(e);`
   - (C) `l.remove();`
   - (D) `l.insert(e);`
   - (E) None
5. Line 5:
   - (A) **if** `(l.empty())`
   - (B) **if** `(!l.full())`
   - (C) **if** `(!l.last())`
   - (D) **if** `(!l.empty())`
   - (E) None
6. Line 6:
   - (A) **while** `(!l.full())`
   - (B) **while** `(!l.last())`
   - (C) **if** `(!l.last())`
   - (D) **while** `(l.last())`
   - (E) None
7. Line 7:
   - (A) `l.findFirst();`
   - (B) `l.insert(e);`
   - (C) `l.findNext();`
   - (D) `l.remove();`
   - (E) None
8. Line 8:

(A) `l.update(e); }}`

(B) `l.insert(l.retrieve()); }}`

(C) `l.remove(); }}`

(D) `l.update(l.retrieve()); }}`

(E) None

Question 2 ..... 10 points

Choose the correct answer (the element in **bold** is the current element):

```
private static <T> void f(DoubleLinkedList<T> l1, DoubleLinkedList<T> l2) {
  if (l1.last() || l2.first())
    return;
  if (l1.retrieve().equals(l2.retrieve())) {
    l1.remove();
    l2.remove();
  } else
    l1.findNext();
  l2.findPrevious();
  f(l1, l2);
}
```

1. If l1: **B** A A D B C, and l2: B B D D A **C**, then after calling `f`:

   (A) l1: B A D **C**

   (B) l1: B A **C**

   (C) There is infinite recursion

   (D) The code may throw an exception

   (E) None

2. If l1: **A** A B B, and l2: B B A **A**, then after calling `f`:

   (A) l1 becomes empty

   (B) l1: **B**

   (C) There is infinite recursion

   (D) The code may throw an exception

   (E) None

7. Line 7:
   - (A) `while(t.length()!= 0){`
   - (B) `while(t.top != null){`
   - (C) `while(!t.empty()){`
   - (D) `while(!t.last()){`
   - (E) None

8. Line 8:
   - (A) `T e = t.serve();`
   - (B) `T e = t.pop();`
   - (C) `T e = t.top.data;`
   - (D) `T e = t.retrieve();`
   - (E) None

9. Line 9: −2
   - (A) `st2.push(e); x.push(e);}`
   - (B) `t.push(e); st2.push(e);}`
   - (C) `st1.push(e); t.push(e);}`
   - (D) `st2.push(e); st1.push(e);}`
   - (E) None

10. Line 10: −2
    - (A) `while(x.length()!= 0)`
    - (B) `while(q.next!=null)q = q.next;`
    - (C) `while(!x.empty())`
    - (D) `while(!x.last())`
    - (E) None

    while(!x.empty)
    x.push(x.pop);

11. Line 11: −2
    - (A) `q.next = r;`
    - (B) `st1.push(x.pop());`
    - (C) `st1.push(x.retrieve());`
    - (D) `st1.push(x.serve());`
    - (E) None

Question 2 ..... 10 ..... 10 points

Consider the function `f` below:

```
public static <T> void f(Queue<T> q, T e) {
  T temp;
  int count = q.length();
  for (int i = 0; i < count; i++) {
    temp = q.serve();
    if (!temp.equals(e))
      q.enqueue(temp);
 }}
```

Choose the correct result in each of the following cases:

1. The queue `q`: A, B, C, C, D, E, after calling `f(q,'C')`, `q` becomes:
   - (A) A, B, D, E
   - (B) $C, C,$
   - (C) E, D, B, A
   - (D) D, B, A
   - (E) None

2. The queue `q`: A, B, C, D, E, after calling `f(q,'F')`, `q` becomes:
   - (A) $A, A, B, B, C, C, D, D, E, E$
   - (B) E, B, C, D, A
   - (C) B, C, D, E
   - (D) A, B, C, C, D, E
   - (E) None

Exam 2 - MoveAfter

Q1

Line1	stack<T> temp = new LinkedStack<T>(); stack<T> x = new LinkedStack<T>()
Line2	for(int j = 0; j<=i; j++)
Line3	temp.push(st1.pop());
Line4	while(!st2.empty())
Line5	x.push(st2.pop());
Line6	while(!x.empty())
Line7	T e = x.pop();
Line8	st1.push(e); st2.push(e)
Line9	while(!temp.empty())
Line10	st1.push(temp.pop());
Line11	None: ~~while~~

Q2 - 1

4, 7, 2, 5, 5, 1

Q2 - 2

None

15, 20, 35, 15, 20

Sequential number:		Section/Instructor: 31205
1	2	Total/40
25	0	25/40

Question 1 ..... 25 ..... 30 points

Write the method **public static** `<T> void moveAfter(Stack<T> st1, Stack<T> st2, int i)` (user of the Stack ADT), that takes as input two stacks st_1, st_2 and an index i. It moves the elements of stack st_2 after the element at position i in stack st_1. Assume that i is within the range of stack st_1, and that the top element has an index 0. The stack st_2 must **not change** after the call.

Example 1. *If st_1 (top to bottom): 5, 2, 4, 1 and st_2 (top to bottom): 8, 9. After calling* `moveAfter(st1, st2, 1)`, *st_1 will be (top to bottom): 5, 2, 8, 9, 4, 1.*

1. Line 1:
   - (A) `Stack<T> temp = new LinkedStack<T>();`
   - (B) `Queue<T> temp = new LinkedQueue<T>();`
   - (C) `List<T> temp = new LinkedList<T>();`
   - (D) `Node<T> n1 = st1.top, n2 = st2.top;`
   - (E) None
2. Line 2:
   - (A) `for(int j = 0; j < i; j++)`
   - (B) `for(int j = 1; j <= i; j++)`
   - (C) `int c = 0;`
   - (D) `for(int j = 0; j <= i; j++)`
   - (E) None
3. Line 3:
   - (A) `temp.push(st1.pop());`
   - (B) `temp.enqueue(st1.pop());`
   - (C) `while(n1 != null){`
   - (D) `temp.insert(st1.pop());`
   - (E) None
4. Line 4:
   - (A) `Stack<T> x = st2;`
   - (B) `Queue<T> x = new LinkedQueue<T>();`
   - (C) `Stack<T> x = new LinkedStack<T>();`
   - (D) `Stack<T> x = new ArrayStack(10);`
   - (E) None
5. Line 5:
   - (A) `while(!st1.full())`
   - (B) `while(!st2.empty())`
   - (C) `while(!x.full())`
   - (D) `while(st2.top != null)`
   - (E) None
6. Line 6:
   - (A) `x.push(st2.pop());`
   - (B) `x.push(st1.pop());`
   - (C) `x.enqueue(st2.pop());`
   - (D) `st2.push(x.pop());`
   - (E) None

7. Line 7:

   (A) `while(!x.last()){`

   (B) `while(x.length()!= 0){`

   (C) `while(!x.empty()){`

   (D) `while(!st1.empty()){`

   (E) None

8. Line 8:

   (A) `T e = st2.pop();`

   (B) `T e = x.pop();`

   (C) `T e = x.top.data;`

   (D) `T e = st1.pop();`

   (E) None

9. Line 9: -2

   (A) `st1.push(e); st2.push(e);}`

   (B) `x.push(e); st2.push(e);}`

   (C) `st2.push(e); temp.push(e);}`

   (D) `st1.push(e); x.push(e);}`

   (E) None

10. Line 10:

   (A) `while(temp.length()!= 0)`

   (B) `c++;}`

   (C) `while(!temp.empty())`

   (D) `while(!temp.last())`

   (E) None

11. Line 11:

   (A) `n1.next = new Node<T>(n2.data);`

   (B) `st1.push(temp.top.data);`

   (C) `st1.push(temp.serve());`

   (D) `st1.push(temp.retrieve());`

   (E) None

Question 2 ..... 0 ..... 10 points

Consider the function f below:

```
public static <T> void f(Queue<T> q1, Queue<T> q2) {
  int countQ1 = q1.length();
  int countQ2 = q2.length();
  for (int i = 0; i < countQ1; i++)
    q2.enqueue(q1.serve());
  for (int i = 0; i < countQ2; i++)
    q1.enqueue(q2.serve());
}
```

Choose the correct result in each of the following cases:

1. The queue `q1`: $2, 5, 1, 5, 2$ and `q2`: $4, 7, 2, 5, 5, 1$, after calling `f(q1,q2)`, `q1` becomes:

   (A) $2, 5, 1, 5, 2$

   (B) $2, 5, 1, 5, 2, 4, 7, 2, 5, 5, 1$ -5

   (C) $4, 7, 2, 5, 5, 1$

   (D) $4, 7, 2, 5, 5, 1, 2, 5, 1, 5, 2$

   (E) None

2. The queue `q1`: $10, 20, 30, 20, 40$ and `q2`: $15, 20, 35, 15, 20$, after calling `f(q1,q2)`, `q1` becomes:

   (A) $10, 20, 30, 20, 40$ -5

   (B) $10, 20, 30, 20, 40, 15, 20, 35, 15, 20$

   (C) $20, 15, 35, 20, 15$

   (D) $15, 20, 35, 15, 20, 10, 20, 30, 20, 40$

   (E) None

Exam1 – MoreAfter version 1

1. (E) stack<T> x = new LinkedStack<T>();
2. (B) for(int j=0; j<=i; j++)
3. (B) x.push(st1.pop());
4. (B) stack<T> t = new LinkedStack<T>();
5. (D) while(!st2.empty());
6. (B) t.push(st2.pop());
7. (C) while(!t.empty()){
8. (B) T e = t.pop();
9. (D) st2.push(e); st1.push(e); }
10. (C) while(!x.empty())
11. (B) st1.push(x.pop());

P2

0508793242

---

1. A B ~~C~~ ~~C~~ D E

(A) A, B, D, E

2. (E) None A, B, C, D, E

1	2	Total/40
19	0	19/40

Question 1 ..... 19 ..... 30 points

Write the method **public boolean** `checkTotalTop(Stack<Integer> s)`, user of ADT Stack, that returns *true* if the top element of a stack of Integers is equal to the total sum of all lower elements in the stack, *false* otherwise. The stack `s` should not be changed after the call. Assume `s` is not empty.

Example 1. *if `s` (top to bottom): 7, 4, 2, 1 then `checkTotalTop(s)` returns true, whereas if `s` (top to bottom): 11, 3, 5, 6 then `checkTotalTop(s)` returns false.*

total = 7 x = 7

1. Line 1:
   - (A) `Stack<Integer>t=new LinkedStack<Integer>();`
   - (B) `Node<T> p = s.top.next;`
   - (C) `List<Integer>t=new LinkedList<Integer>();`
   - (D) `Queue<Integer>t=new LinkedQueue<Integer>();`
   - (E) None
2. Line 2:
   - (A) `int x = s.push();`
   - (B) `int x = s.serve();`
   - (C) `int x = s.retrieve();`
   - (D) `int x = s.pop();`
   - (E) None
3. Line 3:
   - (A) `int total = 0;`
   - (B) `int total = x;`
   - (C) `int total = s.pop();`
   - (D) `int total = t.pop();`
   - (E) None
4. Line 4:
   - (A) `while (t.length()!= 0){`
   - (B) `while (p != null){`
   - (C) `while (!s.empty()){`
   - (D) `while (s.size != 0){`
   - (E) None
5. Line 5: −3
   - (A) `int e = t.serve();`
   - (B) `total += s.pop()`
   - (C) `total = s.pop();`
   - (D) `int e = s.pop()`
   - (E) None
6. Line 6: −3
   - (A) `total += t.serve();`
   - (B) `total += e;`
   - (C) `total += s.pop();`
   - (D) `total = e;`
   - (E) None
7. Line 7: −3
   - (A) `t.push(e); }`

(B) s.push(e); }

(C) t.enqueue(e); }

(D) t.push(s.pop()); }

(E) None

8. Line 8:

(A) while (!t.last())

(B) total -= t.pop();

(C) while (!s.empty())

(D) while (!t.empty())

(E) None

9. Line 9:

(A) s.enqueue(t.serve());

(B) total -= t.pop();

(C) t.push(s.pop());

(D) s.push(t.pop());

(E) None

10. Line 10: -2

(A) s.pop(x);

(B) s.push(x);

(C) s.push(total);

(D) t.push(s.pop());

(E) None

11. Line 11:

(A) return x == ts.serve();

(B) return s.top.data == total;

(C) return s == total;

(D) return x == total;

(E) None

Question 2 ..... 10 points

Consider the function f below:

```
public static <T> void f(Queue<T> q) {
  for (int i = 0; i < q.length() / 2; i++) {
    T e1 = q.serve(); T e2 = q.serve();
    q.enqueue(e2); q.enqueue(e1);
  }
  if (q.length() % 2 == 1) {
    q.enqueue(q.serve());
  }}
```

Choose the correct result in each of the following cases:

1. The queue q: A, B, C, D, after calling f(q), q becomes:

(A) A, A, B, B, C, C, D, D

(B) A, B, C, D

(C) $D, B, C, A,$ -5

(D) D, C, B, A

(E) None

2. The queue q: A, B, C, D, E, after calling f(q), q becomes:

(A) A, B, C, D, E

(B) E, C, D, A, B -5

(C) E, D, C, B, A

(D) B, A, D, C, E

(E) None

Exam3 – checkTotalTop Q1

1. (A) stack<Integer> t = new LinkedStack<Integer>();
2. (D) int x = s.pop();
3. (A) int total = 0;
4. (C) while(!s.empty()){
5. (B) int e = s.pop();
6. (B) total += e;
7. (A) t.push(e); }
8. (D) while(!t.empty())
9. (D) s.push(t.pop());
10. (B) s.push(x);
11. (D) return x == total

Q2

1. A B C D | B A C D

(E) None [B A C D]

2. A B C D E

(D) B A D C E

1	2	Total

Question 1 ..... 30 points

Write the method `pullUpBottom`, user of ADT Stack, that moves the element in the bottom of the stack if it exists to the top without changing the order of the other elements. The method signature is: `public static <T> void pullUpBottom(Stack<T> st)`.

Example 1. *If st (top to bottom): $A \to B \to E \to C$, after calling* `pullUpBottom(st)`, *st becomes $C \to A \to B \to E$.*

1. Line 1:

A (A) `if (st.size == 0)`

(B) `while (! st.empty())`

(C) `if(st.size == 1)`

(D) `if (st.empty())`

(E) None

2. Line 2:

(A) `return true;`

(B) `return st;`

(C) `return;`

D (D) `return false;`

(E) None

3. Line 3:

(A) `List<T> x = new LinkedList<T>();`

(B) `Queue<T> x = new LinkedQueue<T>();`

C (C) `Stack<T> x = new LinkedStack<T>();`

(D) `Node<T> n = s.top;`

(E) None

4. Line 4:

A (A) `T e = st.pop();`

(B) `T e;`

(C) `T e = n.data;`

(D) `T e = st.top.data;`

(E) None

5. Line 5:

(A) `while(n != null)`

(B) `while(st.empty())`

(C) `while(st.length()!= 0)`

D (D) `while(!st.empty())`

(E) None

6. Line 6:

(A) `x.insert(st.pop());`

(B) `n = n.next;`

(C) `x.push(st.pop());`

(D) `x.enqueue(st.pop());`

E (E) None

7. Line 7:

   (A) `if(x.empty())`

   (B) `x.findFirst();`

   C (C) `if(!x.empty())`

   (D) `if(x.top != null)`

   (E) None

8. Line 8:

   (A) `e = x.retrive();`

   (B) `e = x.serve();`

   (C) `e = x.top.data;`

   D (D) `e = x.pop();`

   (E) None

9. Line 9:

   A (A) `while(!x.empty())`

   (B) `while(!x.last()){`

   (C) `n = x.top;`

   (D) `while(x.length()!= 0)`

   (E) None

10. Line 10:

    (A) `while(n!=null)`

    (B) `st.push(x.serve());`

    C (C) `st.push(x.pop());`

    (D) `st.enqueue(x.retrieve());`

    (E) None

11. Line 11:

    (A) `st.pop(e)`

    B (B) `st.push(e);`

    (C) `st.top.data = e;`

    (D) `st.insert(e);`

    (E) None

Question 2 ..... 10 points

Consider the function `f` below. Choose the correct result in each of the cases below:

```
public <T> void f(Queue<T> q) {
  int n = q.length() / 2;
  for (int i = 1; i <= n; i++) {
    T e = q.serve();
    q.enqueue(q.serve());
    q.enqueue(e);
  }
  if (2 * n != q.length())
    q.enqueue(q.serve());
}
```

n=2

1. The queue `q`: 5, 6, 1, 2, 3, after calling `f(q)`, `q` becomes:

   (A) 5, 6, 1, 2, 3

   (B) 3, 2, 1, 6, 5

   C (C) 6, 5, 2, 1, 3

   (D) 3, 1, 2, 5, 6

   (E) None

2. The queue `q`: 1, 2, 3, 4, after calling `f(q)`, `q` becomes:

   A (A) 1, 1, 2, 2, 3, 3, 4, 4

   (B) 2, 1, 3, 4

   (C) 4, 2, 3, 1,

   (D) 4, 3, 2, 1

   (E) None

Exam4 - pullup Bottom

Q1

Line 1	(D) if(st.empty())
Line 2	(C) return;
Line 3	(C) stack<T>x=new LinkedStack<>();
Line 4	(A) T e = st.pop();
Line 5	(D) while(!st.empty())
Line 6	(C) x.push(st.pop());
Line 7	(C) if(!x.empty())
Line 8	(D) e = x.pop();
Line 9	(A) while(!x.empty())
Line 10	(C) st.push(x.pop());
Line 11	(B) st.push(e);

Examy تابع

2*2!=5

Q2

① i=1 i=2

5 6 1 2 3 | 6 5 2 1 3

e

5

الاجابة الصحيحة ⓒ 6 5 2 1 3

② 1 2 3 4 | 2 1 4 3

e

1

الاجابة الصحيحة 2 1 4 3

ⓕ None

غير موجودة في الخيارات

③

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	Quiz 3 - Fall 2018

Studen[redacted]

Sequential number: Section/Instruct[redacted]

1	2	Total/40
24	10	34/40

Question 1 ..... 24 ..... 30 points

Write the method **public static <T> void moveAfter(Stack<T> st1, Stack<T> st2, int i)** (user of the Stack ADT), that takes as input two stacks st_1, st_2 and an index i. It moves the elements of stack st_2 after the element at position i in stack st_1. Assume that i is within the range of stack st_1, and that the top element has an index 0. The stack st_2 must **not change** after the call.

Example 1. *If st_1 (top to bottom): 5, 2, 4, 1 and st_2 (top to bottom): 8, 9. After calling* `moveAfter(st1, st2, 1)`, *st_1 will be (top to bottom): 5, 2, 8, 9, 4, 1.*

1. Line 1:
   - (A) `Node<T> p = st1.top, q = st2.top; int c = 0;`
   - (B) `Queue<T> x = new ArrayQueue<T>(100);`
   - (C) `Stack<T> x = new LinkedStack<T>();`
   - (D) `T[] x = new T[100];`
   - (E) None
2. Line 2:
   - (A) `while (!st1.empty)`
   - (B) `for(int j = 0; j <= i; j++)`
   - (C) `while(p != null){`
   - (D) `for(int j = 0; j < i; j++)`
   - (E) None
3. Line 3:
   - (A) `x[j] = st1.pop();`
   - (B) `x.push(st1.pop());`
   - (C) `if(c == i)break;`
   - (D) `x.enqueue(st1.pop());`
   - (E) None
4. Line 4:
   - (A) `Stack<T> t = st2;`
   - (B) `Stack<T> t = new LinkedStack<T>();`
   - (C) `Queue<T> t = new LinkedQueue<T>();`
   - (D) `Queue<T> t = st2;`
   - (E) None
5. Line 5:
   - (A) `while(!st1.empty())`
   - (B) `while(st2.empty())`
   - (C) `while(st2.top != null)`
   - (D) `while(!st2.empty())`
   - (E) None
6. Line 6:
   - (A) `t.push(st2.top.data);`
   - (B) `t.push(st2.pop());`
   - (C) `t.enqueue(st2.pop());`
   - (D) `st2.push(t.pop());`
   - (E) None

7. Line 7:

   (A) `while(!x.last()){`
   (B) `while(x.length()!= 0){`
   (C) `while(!x.empty()){`
   (D) `while(!st1.empty()){`
   (E) None

8. Line 8:

   (A) `T e = st2.pop();`
   (B) `T e = x.pop();`
   (C) `T e = x.top.data;`
   (D) `T e = st1.pop();`
   (E) None

9. Line 9:

   (A) `st1.push(e); st2.push(e);}`
   (B) `x.push(e); st2.push(e);}`
   (C) `st2.push(e); temp.push(e);}`
   (D) `st1.push(e); x.push(e);}`
   (E) None

10. Line 10:

   (A) `while(temp.length()!= 0)`
   (B) `c++; }`
   (C) `while(!temp.empty())`
   (D) `while(!temp.last())`
   (E) None

11. Line 11:

   (A) `n1.next = new Node<T>(n2.data);`
   (B) `st1.push(temp.top.data);`
   (C) `st1.push(temp.serve());`
   (D) `st1.push(temp.retrieve());`
   (E) None

Question 2 ..... 10 points

Consider the function f below:

```
public static <T> void f(Queue<T> q1, Queue<T> q2) {
  int countQ1 = q1.length();
  int countQ2 = q2.length();
  for (int i = 0; i < countQ1; i++)
    q2.enqueue(q1.serve());
  for (int i = 0; i < countQ2; i++)
    q1.enqueue(q2.serve());
}
```

Choose the correct result in each of the following cases:

1. The queue q1: 2, 5, 1, 5, 2 and q2: 4, 7, 2, 5, 5, 1, after calling `f(q1,q2)`, q1 becomes:

   (A) 2, 5, 1, 5, 2
   (B) 2, 5, 1, 5, 2, 4, 7, 2, 5, 5, 1
   (C) 4, 7, 2, 5, 5, 1
   (D) 4, 7, 2, 5, 5, 1, 2, 5, 1, 5, 2
   (E) None

2. The queue q1: 10, 20, 30, 20, 40 and q2: 15, 20, 35, 15, 20, after calling `f(q1,q2)`, q1 becomes:

   (A) 10, 20, 30, 20, 40
   (B) 10, 20, 30, 20, 40, 15, 20, 35, 15, 20
   (C) 20, 15, 35, 20, 15
   (D) 15, 20, 35, 15, 20, 10, 20, 30, 20, 40
   (E) None

Consider the function f below:

```
public static <T> void f(Queue<T> q1, Queue<T> q2) {
  int countQ1 = q1.length();
  int countQ2 = q2.length();
  for (int i = 0; i < countQ1; i++)
    q2.enqueue(q1.serve());
  for (int i = 0; i < countQ2; i++)
    q1.enqueue(q2.serve());
}
```

Choose the correct result in each of the following cases:

1. The queue q1: 2, 5, 1, 5, 2 and q2: 4, 7, 2, 5, 5, 1, after calling f(q1,q2), q1 becomes:

   (A) 2, 5, 1, 5, 2

   (B) 2, 5, 1, 5, 2, 4, 7, 2, 5, 5, 1 -5

   (C) 4, 7, 2, 5, 5, 1

   (D) 4, 7, 2, 5, 5, 1, 2, 5, 1, 5, 2

   (E) None

2. The queue q1: 10, 20, 30, 20, 40 and q2: 15, 20, 35, 15, 20, after calling f(q1,q2), q1 becomes:

   (A) 10, 20, 30, 20, 40

   (B) 10, 20, 30, 20, 40, 15, 20, 35, 15, 20 -5

   (C) 20, 15, 35, 20, 15

   (D) 15, 20, 35, 15, 20, 10, 20, 30, 20, 40

   (E) None

1	2	

Question 1 ..... 30 points

Write the method `pullUpBottom`, user of ADT Stack, that moves the element in the bottom of the stack if it exists to the top without changing the order of the other elements. The method signature is: `public static <T> void pullUpBottom(Stack<T> st)`.

Example 1. *If st (top to bottom): $A \to B \to E \to C$, after calling* `pullUpBottom(st)`, *st becomes $C \to A \to B \to E$.*

1. Line 1:

A (A) `if (st.size == 0)`

(B) `while () st.empty())`

(C) `if(st.size == 1)`

(D) `if (st.empty())`

(E) None

2. Line 2:

(A) `return true;`

(B) `return st;`

(C) `return;`

D (D) `return false;`

(E) None

3. Line 3:

(A) `List<T> x = new LinkedList<T>();`

(B) `Queue<T> x = new LinkedQueue<T>();`

C (C) `Stack<T> x = new LinkedStack<T>();`

(D) `Node<T> n = s.top;`

(E) None

4. Line 4:

A (A) `T e = st.pop();`

(B) `T e;`

(C) `T e = n.data;`

(D) `T e = st.top.data;`

(E) None

5. Line 5:

(A) `while(n != null)`

(B) `while(st.empty())`

(C) `while(st.length()!= 0)`

D (D) `while(!st.empty())`

(E) None

6. Line 6:

(A) `x.insert(st.pop());`

(B) `n = n.next;`

(C) `x.push(st.pop());`

(D) `x.enqueue(st.pop());`

E (E) None

7. Line 7:

(A) `if(s.empty())`

(B) `s.findPyrest();`

C (C) `if(!s.empty())`

(D) `if(s.top != null)`

(E) None

8. Line 8:

(A) `e = s.retrieve();`

(B) `e = s.serve();`

(C) `e = s.top.data;`

C (D) `e = s.pop();`

(E) None

9. Line 9:

A (A) `while(!s.empty())`

(B) `while(!s.last()){`

(C) `e = s.top;`

(D) `while(s.length()!= 0)`

(E) None

10. Line 10:

(A) `while(s!=null)`

(B) `st.push(s.serve());`

C (C) `st.push(s.pop());`

(D) `st.enqueue(s.retrieve());`

(E) None

11. Line 11:

(A) `st.pop(e)`

B (B) `st.push(e);`

(C) `st.top.data = e;`

(D) `st.insert(e);`

(E) None

Question 2 ..... 10 points

Consider the function f below. Choose the correct result in each of the cases below:

```
public <T> void f(Queue<T> q) {
  int n = q.length() / 2;
  for (int i = 1; i <= n; i++) {
    T e = q.serve();
    q.enqueue(q.serve());
    q.enqueue(e);
  }
  if (2 * n != q.length())
    q.enqueue(q.serve());
}
```

1. The queue q: 5, 6, 1, 2, 3, after calling f(q), q becomes:

(A) 5, 6, 1, 2, 3

(B) 3, 2, 1, 6, 5

C (C) 6, 5, 2, 1, 3

(D) 3, 1, 2, 5, 6

(E) None

2. The queue q: 1, 2, 3, 4, after calling f(q), q becomes:

A (A) 1, 1, 2, 2, 3, 3, 4, 4

(B) 2, 1, 3, 4

(C) 4, 2, 3, 1

(D) 4, 3, 2, 1

(E) None

A B D C

C D B A

C A B D

7. Line 7:

   (A) if(x.empty()
   (B) x.findFirst();
   (C) if(!x.empty())
   (D) if(x.top != null)
   (E) None

8. Line 8:

   (A) e = x.retrive();
   (B) e = x.serve();
   (C) e = x.top.data;
   (D) e = x.pop();
   (E) None

9. Line 9:

   (A) while(!x.empty())
   (B) while(!x.last()){
   (C) n = x.top;
   (D) while(x.length()!= 0)
   (E) None

10. Line 10:

   (A) while(n!=null)
   (B) st.push(x.serve());
   (C) st.push(x.pop());
   (D) st.enqueue(x.retrieve());
   (E) None

11. Line 11:

   (A) st.pop(e)
   (B) st.push(e);
   (C) st.top.data = e;
   (D) st.insert(e);
   (E) None

Question 2 ..... 10 points

Consider the function f below. Choose the correct result in each of the cases below:

```
public <T> void f(Queue<T> q) {
  int n = q.length() / 2;
  for (int i = 1; i <= n; i++) {
    T e = q.serve();
    q.enqueue(q.serve());
    q.enqueue(e);
  }
  if (2 * n != q.length())
    q.enqueue(q.serve());
}
```

n = 2

1, 2 ✓

6123 → 2365

1236 → 3652

12365 → 36521

65213

1. The queue q: 5, 6, 1, 2, 3, after calling f(q), q becomes:

   (A) 5, 6, 1, 2, 3
   (B) 3, 2, 1, 6, 5
   (C) 6, 5, 2, 1, 3
   (D) 3, 1, 2, 5, 6
   (E) None

2. The queue q: 1, 2, 3, 4, after calling f(q), q becomes:

   (A) 1, 1, 2, 2, 3, 3, 4, 4
   (B) 2, 1, 3, 4
   (C) 4, 2, 3, 1,
   (D) 4, 3, 2, 1
   (E) None

Sequential number:		Section/Instructor: 31205
1	2	Total/40
25	0	25/40

Question 1 ..... 25 ..... 30 points

Write the method **public static <T> void moveAfter(Stack<T> st1, Stack<T> st2, int i)** (user of the Stack ADT), that takes as input two stacks st_1, st_2 and an index i. It moves the elements of stack st_2 after the element at position i in stack st_1. Assume that i is within the range of stack st_1, and that the top element has an index 0. The stack st_2 must **not change** after the call.

Example 1. *If st_1 (top to bottom): 5, 2, 4, 1 and st_2 (top to bottom): 8, 9. After calling* `moveAfter(st1, st2, 1)`, *st_1 will be (top to bottom): 5, 2, 8, 9, 4, 1.*

1. Line 1:
   - (A) `Stack<T> temp = new LinkedStack<T>();`
   - (B) `Queue<T> temp = new LinkedQueue<T>();`
   - (C) `List<T> temp = new LinkedList<T>();`
   - (D) `Node<T> n1 = st1.top, n2 = st2.top;`
   - (E) None
2. Line 2:
   - (A) `for(int j = 0; j < i; j++)`
   - (B) `for(int j = 1; j <= i; j++)`
   - (C) `int c = 0;`
   - (D) `for(int j = 0; j <= i; j++)`
   - (E) None
3. Line 3:
   - (A) `temp.push(st1.pop());`
   - (B) `temp.enqueue(st1.pop());`
   - (C) `while(n1 != null){`
   - (D) `temp.insert(st1.pop());`
   - (E) None
4. Line 4:
   - (A) `Stack<T> x = st2;`
   - (B) `Queue<T> x = new LinkedQueue<T>();`
   - (C) `Stack<T> x = new LinkedStack<T>();`
   - (D) `Stack<T> x = new ArrayStack(10);`
   - (E) None
5. Line 5:
   - (A) `while(!st1.full())`
   - (B) `while(!st2.empty())`
   - (C) `while(!x.full())`
   - (D) `while(st2.top != null)`
   - (E) None
6. Line 6:
   - (A) `x.push(st2.pop());`
   - (B) `x.push(st1.pop());`
   - (C) `x.enqueue(st2.pop());`
   - (D) `st2.push(x.pop());`
   - (E) None

1	2	3

Question 1 ..... 20 points

Write the method `pullUpBottom`, user of ADT Stack, that moves the element in the bottom of the stack if it exists to the top without changing the order of the other elements. The method signature is: `public static <T> void pullUpBottom(Stack<T> st)`.

Example 1. *If st (top to bottom): $A \to B \to E \to C$, after calling* `pullUpBottom(st)`, *st becomes $C \to A \to B \to E$.*

1. Line 1:

A (A) `if (st.size == 0)`
(B) `while (! st.empty())`
(C) `if(st.size == 1)`
(D) `if (st.empty())`
(E) None

2. Line 2:

(A) `return true;`
(B) `return st;`
(C) `return;`
D (D) `return false;`
(E) None

3. Line 3:

(A) `List<T> x = new LinkedList<T>();`
(B) `Queue<T> x = new LinkedQueue<T>();`
C (C) `Stack<T> x = new LinkedStack<T>();`
(D) `Node<T> n = s.top;`
(E) None

4. Line 4:

A (A) `T e = st.pop();`
(B) `T e;`
(C) `T e = n.data;`
(D) `T e = st.top.data;`
(E) None

5. Line 5:

(A) `while(n != null)`
(B) `while(st.empty())`
(C) `while(st.length()!= 0)`
D (D) `while(!st.empty())`
(E) None

6. Line 6:

(A) `s.insert(st.pop());`
(B) `n = n.next;`
(C) `s.push(st.pop());`
(D) `s.enqueue(st.pop());`
E (E) None

7. Line 7:

   (A) `if(x.empty())`

   (B) `x.findFirst();`

   C (C) `if(!x.empty())`

   (D) `if(x.top != null)`

   (E) None

8. Line 8:

   (A) `e = x.retrive();`

   (B) `e = x.serve();`

   (C) `e = x.top.data;`

   D (D) `e = x.pop();`

   (E) None

9. Line 9:

   A (A) `while(!x.empty())`

   (B) `while(!x.last()){`

   (C) `n = x.top;`

   (D) `while(x.length()!= 0)`

   (E) None

10. Line 10:

    (A) `while(n!=null)`

    (B) `st.push(x.serve());`

    C (C) `st.push(x.pop());`

    (D) `st.enqueue(x.retrieve());`

    (E) None

11. Line 11:

    (A) `st.pop(e)`

    B (B) `st.push(e);`

    (C) `st.top.data = e;`

    (D) `st.insert(e);`

    (E) None

Question 2 ..... 10 points

Consider the function `f` below. Choose the correct result in each of the cases below:

```
public <T> void f(Queue<T> q) {
  int n = q.length() / 2;          n=2
  for (int i = 1; i <= n; i++) {
    T e = q.serve();
    q.enqueue(q.serve());
    q.enqueue(e);
  }
  if (2 * n != q.length())
    q.enqueue(q.serve());
}
```

1. The queue `q`: 5, 6, 1, 2, 3, after calling `f(q)`, `q` becomes:

   (A) 5, 6, 1, 2, 3

   (B) 3, 2, 1, 6, 5

   C (C) 6, 5, 2, 1, 3

   (D) 3, 1, 2, 5, 6

   (E) None

2. The queue `q`: 1, 2, 3, 4, after calling `f(q)`, `q` becomes:

   A (A) 1, 1, 2, 2, 3, 3, 4, 4

   (B) 2, 1, 3, 4

   (C) 4, 2, 3, 1,

   (D) 4, 3, 2, 1

   (E) None

(B) s.push(e); }

(C) t.enqueue(e); }

(D) t.push(s.pop()); }

(E) None

8. Line 8:

(A) while (!t.last())

(B) total -= t.pop();

(C) while (!s.empty())

(D) while (!t.empty())

(E) None

9. Line 9:

(A) s.enqueue(t.serve());

(B) total -= t.pop();

(C) t.push(s.pop());

(D) s.push(t.pop());

(E) None

10. Line 10: -2

(A) s.pop(x);

(B) s.push(x);

(C) s.push(total);

(D) t.push(s.pop());

(E) None

11. Line 11:

(A) return x == ts.serve();

(B) return s.top.data == total;

(C) return s == total;

(D) return x == total;

(E) None

Question 2 ..... 0 ..... 10 points

Consider the function f below:

```
public static <T> void f(Queue<T> q) {
  for (int i = 0; i < q.length() / 2; i++) {
    T e1 = q.serve(); T e2 = q.serve();
    q.enqueue(e2); q.enqueue(e1);
  }
  if (q.length() % 2 == 1) {
    q.enqueue(q.serve());
  }}
```

Choose the correct result in each of the following cases:

1. The queue q: A, B, C, D, after calling f(q), q becomes:

(A) A, A, B, B, C, C, D, D

(B) A, B, C, D

(C) D, B, C, A, -5

(D) D, C, B, A

(E) None

2. The queue q: A, B, C, D, E, after calling f(q), q becomes:

(A) A, B, C, D, E

(B) E, C, D, A, B -5

(C) E, D, C, B, A

(D) B, A, D, C, E

(E) None

King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Final Exam - Spring 2020**
Date: 02/05/2020	Duration: 3 hours

Do not answer in this sheet and do not submit it.

Question 1 ..... 20 points

(a) Choose the correct frequency for every line as well as the total O of the following code:

```
1 sum = 1;
2 for (i = 1; i <= n; i++) {
3   sum+= i;
4   for (j = i; j >= 2; j--)
5     sum--;}
```

1. Line 1: (A) 1 (B) 2 (C) 0 (D) n (E) $2n$
2. Line 2: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
3. Line 3: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
4. Line 4: (A) n^2 (B) $n(n-1)/2$ (C) $(2n+1)/2$ (D) $(2n-1)/2$ (E) $n(n+1)/2$
5. Line 5: (A) n^2 (B) $n(n-1)/2$ (C) $(2n+1)/2$ (D) $(2n-1)/2$ (E) $n(n+1)/2$
6. Total O: (A) 1 (B) n (C) n^2 (D) $n\log(n)$ (E) n^3

(b) Choose the correct frequency for every line as well as the total O of the following code:

```
1 count = 0;
2 for (i = 1; i < n+1; i++)
3   count ++;
4 for (j = 0; j <= count; j++)
5   k = j+1;
```

1. Line 1: (A) 0 (B) 1 (C) 2 (D) n (E) n^2
2. Line 2: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
3. Line 3: (A) n (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n-2$
4. Line 4: (A) $count+1$ (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n(n+1)/2$
5. Line 5: (A) $count+1$ (B) $n+1$ (C) $n-1$ (D) $n+2$ (E) $n(n-1)/2$
6. Total O: (A) 1 (B) n (C) n^2 (D) $n\log(n)$ (E) n^3

(c) Choose the correct frequency for every line as well as the total O of the following code:

```
1 int i = 1;
2 while (i < n) {
3   i++;
4   if (i > 7) break;
5 }
```

1. Line 1: (A) 1 (B) 0 (C) i (D) n (E) $n+1$
2. Line 2: (A) 8 (B) 7 (C) n (D) $n-1$ (E) $n+1$
3. Lines 3: (A) n (B) $n-1$ (C) 6 (D) 7 (E) 8
4. Lines 4: (A) n (B) $n-1$ (C) 6 (D) 7 (E) 8
5. Tightest Total O: (A) 1 (B) n (C) $\log(n)$ (D) n^2 (E) 2^n

(d) Choose the correct answer:

1. $n^7 + n^4 + n^2 + \log n$ is : (A) $O(n^2)$ (B) $O(n^4)$ (C) $O(n^7)$ (D) $O(\log(n))$ (E) None
2. $2^n + n!$ is : (A) $O(n^2)$ (B) $O(2^n)$ (C) $O(n!)$ (D) $O(n^n)$ (E) None
3. $n + \log n^3 + 6$ is : (A) $O(n)$ (B) $O(\log n^3)$ (C) $O(n \log n)$ (D) $O(n^3)$ (E) None
4. The time complexity of inserting an element in a heap of n elements is:
   (A) $O(n^2)$ (B) $O(n)$ (C) $O(2^n)$ (D) $O(\log(n))$ (E) None
5. $n^2 + n \log n^4$ is : (A) $O(n)$ (B) $O(n^2)$ (C) $O(n \log(n))$ (D) $O(n^4)$ (E) None
6. $n^2 + 1000n$ is : (A) $O(n)$ (B) $O(n^2)$ (C) $O(n \log(n))$ (D) $O(nn^2)$ (E) None
7. $n^4 \log n + n!$ is : (A) $O(n!)$ (B) $O(n^4)$ (C) $O(n^5)$ (D) $O(\log(n))$ (E) None
8. Algorithm A is $O(n)$, and Algorithm B is $O(2n)$. Given the same input:
   (A) A always finishes before B. (B) B always finishes before A. (C) A and B finish at the same time. (D) B requires double the time taken by A. (E) None

Question 2 ..... 20 points

(a) Write the method `removeOddElems`, member of the class `LinkedList`, that removes all the elements having an odd position (the position of the first element is 0). The method signature is: **`public void`** `removeOddElems()`.

Example 1. *If* $l : A \to B \to C \to D \to E$, *then* `l.oddElems()` *returns:* $A \to C \to E$.

1. Line 1:
   - (A) **`while`** `(head != null)`
   - (B) **`if`** `(head.next != null)`
   - (C) **`if`** `(head == null)`
   - (D) **`if`** `(head != null)`
   - (E) None
2. Line 2:
   - (A) **`return`** `head;`
   - (B) **`return false`**`;`
   - (C) **`return null`**`;`
   - (D) **`return`** `current;`
   - (E) None
3. Line 3:
   - (A) `Node<T> p = head;`
   - (B) `Node<T> p = head.next;`
   - (C) `Node<T> p = current;`
   - (D) `Node<T> p = head.next.next;`
   - (E) None
4. Line 4:
   - (A) **`while`** `(p.next != null){`
   - (B) **`while`** `(p != null){`
   - (C) **`while`** `(p.next.next != null){`
   - (D) **`while`** `(current.next != null){`
   - (E) None
5. Line 5:
   - (A) `p.next = p;`
   - (B) `p = p.next;`
   - (C) `p.next = p.next.next;`
   - (D) `current.next = p;`
   - (E) None
6. Line 6:

(A) `p = p.next;}`

(B) `p.next.next = current.next;}`

(C) `p = p.next.next;}`

(D) `p.next.next = p.next;}`

(E) None

(b) Consider the function `f` below, member of `LinkedList`:

```
public void f(int n) {
  Node<T> p = head; Node<T> q =  null;
  for (int i = 0; i < n; i++) {
    q = p;
    p = p.next;
  }
  if (p != null) {
    while (p.next != null)
      p = p.next;
    p.next = head;
    head = q.next;
    q.next = null;
  }
}
```

B

head.next

p.next

C

2

E

B, C, D, E, A

C, B, D, E, A

C, D, E, A

0
1

How many times it enters

Choose the correct result in each of the following cases:

1. The list `l`: A, B, C, D, E, after calling `l.f(2)`, `l` becomes:

   (A) A, B (B) D, E, A, B, C (C) C, D, E, A, B (D) A, D, E, B, C (E) None

2. The list `l`: A, B, C, D, E, after calling `l.f(0)`, `l` becomes:

   (A) *empty* (B) A, B, C, D, E (C) B, C, D, E, A (D) B, C, D, E (E) None

3. The list `l`: A, B, C, D, E, after calling `l.f(5)`, `l` becomes:

   (A) *empty* (B) A, B, C, D, E (C) E, D, C, B, A (D) A, D, E, B, C (E) None

4. The list `l`: A, B, C, D, E, after calling `l.f(1)`, `l` becomes:

   (A) A (B) E, A, B, C, D (C) C, D, E, A, B (D) B, C, D, E, A (E) None

Question 3 ..... 20 points

Write the method **public** `<T>` **boolean** `isReverse(Queue<T> q1, Queue<T> q2)`, user of the ADT `Queue`. The method checks if `q2` is the reverse of `q1`. The two queues must not change after the call.

اقلب وحدة بعدين قارن شي
رجع كلش مكانه

1. Line 1:

   (A) `if (q1.length()== q2.length())`

   (B) `if ((q1.length()== 0)&& (q2.length()==0))return true;`

   (C) `if (q1.length()!= q2.length())return false;`

   (D) `if ((q1.length()== 0 )|| (q2.length()== 0))return true;`

   (E) None

2. Line 2:

   (A) `Stack<T> x1 = new Stack<T>(), x2 = new Stack<T>();`

   (B) `Queue<T> x1 = new LinkedQueue<T>(), x2 = new LinkedQueue<T>();`

   (C) `Stack<T> x1 = new LinkedStack<T>(), x2 = new LinkedStack<T>();`

   (D) `Stack<T> x = new LinkedStack<T>();`

   (E) None

   stacks الشغل على

3. Line 3:

   stack

   (A) `while (!q1.empty())`

   queue

   (B) `while (q1.length()!= 0)`

   (C) `while (x1.length()!= 0)`

   (D) `while (!x.full ())`

   (E) None

4. Line 4:

   (A) `x.push(q1.retrieve());`

   (B) `x1.push(current.data);`

   (C) `x1.push(q1.serve());`

   (D) `x1.enquque(q1.serve());`

   (E) None

5. Line 5:

   (A) `while (!x.empty()){`

   (B) `while (x1.size()>0){`

   (C) `boolean eq = false;`

   (D) `boolean eq = true;`

   (E) None

6. Line 6:

   (A) `T e1 = x1.pop(), e2 = x2.pop();`

(B) `T e1 = x.pop(), e2 = q2.serve();`

(C) **while** `(!x1.empty()){`

(D) **while** `(x1.size()>0){`

(E) None

7. Line 7:

(A) **if** `(!e1.equals(e2))`**break**;

(B) `T e1 = x1.pop(), e2 = q1.serve();`

(C) **if** `(!e1.equals(e2))`

(D) `T e1 = x1.pop(), e2 = q2.serve();`

(E) None

8. Line 8:

(A) **if** `(!e1.equals(e2))`**continue**;

(B) **if** `(!e1.equals(e2))eq =` **false**;

(C) `x2.push(e1); q2.enqueue(e1); }`

(D) **return false**;

(E) None

9. Line 9:

(A) `x.push(e1); q2.enqueue(e2); }`

(B) `x2.push(e1); q2.enqueue(e2); }`

(C) `x1.push(e1); q2.enqueue(e1); }`

(D) **return false**

(E) None

10. Line 10:

(A) **while** `(!q2.empty())`

(B) **while** `(!x2.empty())`

(C) **while** `(!q1.full())`

(D) **while** `(!x.last())`

(E) None

11. Line 11:

(A) `q1.enqueue(x1.pop());`

(B) `q1.enqueue(x2.pop());`

(C) `q2.enqueue(x.pop());`

(D) `q2.enqueue(x2.pop());`

(E) None

12. Line 12:

(A) **return false;**

(B) **return true;**

(C) **return** `e1.equals(e2);`

(D) **return** `eq;`

(E) None