MIDTERM EXAM INSTRUCTIONS

1. Midterm Exam: **25 points w/ 0 E.C. points**
2. Due Date & Time: **05-02-2021 at 11:55 PM**

WHAT TO SUBMIT

1. Take-home Exam Report, 1 PDF

HOW TO SUBMIT AND THE RULES TO FOLLOW

- Submit via iLearn, the Assignment and Exam Submission section
- Please follow the exam instructions
- Please follow the Course Policy on Student Conduct and Academic Honesty

PERFORMANCE TRACKER		
ASMT	GRADE	YOUR GRADE
ZOOM	05	
01	20	
02-PREPARATION	25	
02	75	
03	75	
MIDTERM EXAM 01	25	
04	75	
MIDTERM EXAM 02	25	
TOTAL	325	

A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on iLearn.

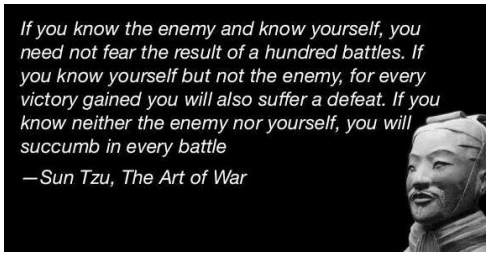
ABOUT

The goal of this take-home exam is for us to **know what we do not know**.

We are taking this exam as seriously as how we take an actual exam in class. Please,

1. Follow all the rules and the guidelines listed at the top of page 1 and page 2
2. Read each question carefully before answering

We will go through the answers to all the exam questions together in class.



STEP A – Take the Exam, **10 points**

1. Allocate 50 quiet minutes to take the exam on page 2 to the last page.
2. Record the date and time when you start.
3. Stop right at minute 50. Record the date and time when you stop the exam.

STEP B – Correct Your Answers, **10 points**

1. Review the related course materials and write code when necessary to find a correct answer for each question. We should be able to find all the answers using the packages, the in-class discussions, our assignments, and the other course materials.
2. At the end of each of your oringal answers, type in *italic* text and:
   - Give your orginal answer a score.
   - List all the mistakes then explain why, you think, you made the mistakes. Add the correct answer you found. Document how you found the correct answer. Document where you found the materials which support the answer.
   - If you did not make any mistakes, please document how you verified that your answer was accurate. Document how you found the correct answer. Document where you found the materials which support the answer. Outline how you could have done better.
   - Record your total score out of 100 points for all the orginal answers.

STEP C – Reflect and Retake the Exam, **5 points**

1. **Problem Solving**: Reflect if you managed the exam time efficiently and if you strategized your test-taking successfully.
2. Repeat steps A to C again if necessary. Please keep appending new contents as directed in Step B.2.
3. Think if the same topics will be tested again in our final exam, what questions we may get.

It is a good idea to do every step of this assignment thoroughly. We are creating a set of materials which we will use to review for the final exam. And this is also the best way to prepare ourselves to succeed in the second half of the semester. Thank you.

1. Section, Date and Time:
   CSC 220.02+03, Due ##-##-#### at ##:## PM

2. Midterm Exam (2 exams, 0 dropped): 100 points

Full Name in **CAPITAL LETTERS** | **SFSU ID**

3. To prepare for this exam, please review all the related materials including WEEK 01-16 packages, slides, mock-up exam(s), reading assignments, in-class practices, sample programs posted in the File Manager, and assignments.
4. You do not need to print this exam. No paper. No handwriting. No scanning. Please type up all your answers in the answer space available in the exam. The provided exam will be in Microsoft Word format. Please submit a single PDF via iLearn.
5. All the rules of an actual exam apply to this exam such as: closed books, closed notes, and no communication with anyone except the course instructor. The course instructor will be available on Zoom or email during the exam time: zoom.ducta.net
6. Please ask all your questions, if any, during the review sessions. Thank you.

HONOR CODE:

- Please follow the CS Department's policies: https://cs.sfsu.edu/student-policies

- Please follow the course's policies: http://csc220.ducta.net/00-README-StudentConduct_AcademicHonesty.pdf

---

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**I**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122

PART A – 50 Points

A.1 - 5 Points
What does **`compareTo()`** possibly return when the method compares **`x`** vs. **`y`**?

A.2 - 5 Points
Among **stack**, **queue**, **deque**, and **priority queue**, which structure(s) does not accept **`null`**? Explain why?

A.3 - 10 Points
What is the output? And **why**?

```
String x, y;
char w;

x = "ComputingQuantum";
y = "ComputingClassical";

System.out.println(y.compareTo(x));
```

Output:
And why:

```
x = "QuantumComputing";
y = "QuantumKomputing";

System.out.println(x.compareTo(y));
```

Output:
And why:

```
x = "ClassicalComputing";
y = "ClassicalComputingX";

System.out.println(y.compareTo(x));
```

Output:
And why:

```
x = "ConventionalComputing";
y = "ConventionalComputinG";

System.out.println(x.compareTo(y));
```

Output:
And why:

```
x = "x" + "y";
w = 'x';
System.out.println(x.compareTo(w));
```

Output:
And why:

A.4 - 10 Points
How does **Selection Sort** work? What are the steps?

9 7 5 8 1 7 4 3 2
Show the contents of the array above **each time** a **Selection Sort** changes it while sorting the array into **ascending order**.

A.5 - 10 Points
How does **Insertion Sort** work? What are the steps?

9 1 8 7 6 5 2 3 4
Show the contents of the array above **each time** an **Insertion Sort** changes it while sorting the array into **ascending order**.

A.6 - 10 Points
How does **Shell Sort** work? What are the steps?

9 6 8 7 5 1 3 2 4 5
Show the contents of the array above **each time** a **Shell Sort** changes it while sorting the array into **ascending order**.

PART B – 50 Points

B.1 - 10 Points

What is a **queue**? How does it work? Use an example and diagrams to explain its operations.

Display the **Queue** at the **marked lines**.

```
Queue<Integer> q = new LinkedList<>();
for (int i = 3; i <= 8; i++) {
    q.add(i);
}
q.remove(q.size() - 1);                  // 1
q.offer(q.element() + 3);                // 2
System.out.println(q.contains(9));       // 3
q.add(q.remove());                       // 4
q.offer(q.peek());                       // 5
```

FRONT	
1.	
2.	
3.	*Output:*
4.	
5.	

B.2 - 10 Points
What is a **deque**? How does it work? Use an example and diagrams to explain its operations.

Display the **Deque** at the **marked lines**.

```
Deque d = new LinkedList<>();
d.add("H");
d.addFirst("O");
d.addLast(d.contains("P"));                 // 1
d.offer(d.element());                       // 2
d.offerLast(d.remove());                    // 3
d.offerFirst(d.remove(d.element()));        // 4
d.push(d.remove("E"));                      // 5
```

FRONT	
1.	
2.	
3.	
4.	
5.	

B.3 - 10 Points
What is a **priority queue**? How does it work? Use an example and diagrams to explain its operations.

Display the **Priority Queue** at the **marked lines**.

```
PriorityQueue<String> pq = new PriorityQueue<>();
pq.offer("C");
pq.add("O");
pq.offer("M");
pq.add("P");
pq.offer(String.valueOf(pq.remove(pq.peek())));       // 1
pq.add(String.valueOf(pq.contains("X")));             // 2
pq.add(String.valueOf(pq.contains(pq.remove())));     // 3
pq.offer(pq.remove());                                // 4
pq.add(pq.peek());                                    // 5
```

FRONT

1.	
2.	
3.	
4.	
5.	

B.4 - 10 Points
What is a **list**? How does it work? Use an example and diagrams to explain its operations.

Display the **List** at the **marked lines**.

```
LinkedList li = new LinkedList();
for (int i = 3; i <= 8; i++) {
   li.add(i);
}
li.remove(li.get(2));
li.set(4,li.peekLast());
li.add(li.size()+1);                // 1
li.addFirst(li.get(li.size()-2));   // 2
Collections.sort(li);               // 3
li.add(li.indexOf(3));              // 4
li.addFirst(li.peekFirst());        // 5
```

FRONT

1.	
2.	
3.	
4.	
5.	

B.5 - 5 Points

Given a **queue** `q` of Integer elements, please write code to check if the elements are:

- Not Sorted
- Sorted in Ascending Order
- Sorted in Descending Order

B.6 - 5 Points
Given 2 **lists** of **comparable `String`** elements and of different lengths, please write code to output:

- "*The tail of A is the reverse of B.*" if
  listA: x, y, z, f, a, b, d, **A, B, C, D, E, F**
  listB: **F, E, D, C, B, A**

- "*The tail of A is NOT the reverse of B.*" if not

DO NOT DETACH THIS PAGE FROM YOUR EXAM.
You may get partial points for what you write on this page.