1. Section, Date and Time:
   CSC 220.01 - Due 08-10-2020 at 11:55 PM

2. Final Exam (1 exam, 0 dropped): 150 points

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

KAYVAUN KHOSHKHOU | 920357344

3. To prepare for this exam, please review all the related materials including WEEK 01-10 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 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 – 65 Points

A.1 - 5 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

How is **`compareTo()`** different from **`equals()`** and **`==`**

compareTo() is used to compare two strings in their lexicography. Each character within a string has a Unicode value which is what the method comparison is based on. Equals() compares two strings and returns either true or false based on whether or not the strings are equal. The main difference between these two is that compareTo() returns an integers and can tell you whether or not the two subjects in comparison are less than each other, equal to each other, or greater than one another. Equals() is boolean so you will only get results back in true or false whether they are equal or not. Lastly, "==" is an equality operator used to check primitive types like Boolean, int, and float. Notice these are datatypes, not objects necessarily, and that's where the main difference lies. With equals(), you are primarily using it to compare objects. Equals() goes more in depth into an object and will check for things like similar strings throughout for example a domain object. You can use it over a more widespread purpose whereas "==" is used for checking equality between two primitives.

A.2 - 5 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

A buffet restaurant asks you to recommend a data structure to manage the lines at their food stations. Which one will you use? Please explain why. [ ] **Stack** [ ] **Queue** [ ] **Deque** [ ] **Priority Queue**

In my opinion, when dining at a buffet there should be a first in first out rule. Especially when it comes to the food stations, if there are people waiting for a dish to be ready to serve, the last person to arrive for the dish should not be served first. Therefore I believe **Queue** would work best here. Priority queue wouldn't be as suited as queue because that would be like having VIP customers at a buffet. When in reality you want everyone to be treated equally. There might be some buffets that offer something like this but for the most part Queue would work better in a general setting. Stack is not suited for this either because it would be as if the chef is hanging out food randomly to the crowd instead of those who have lined up for a particular dish. Dequeue isn't suited because why would the people at the end of the line be served first rather than those who have been waiting at the front?

My conclusion is that **Queue** data structure would work the best out of the given in order to manage the lines at food stations at a buffet.

A.3 - 5 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

What is the output? And **why**?

```
String a, b;
String c = new String("Sorrento");

a = "HashCode";
b = "HashTable";
System.out.println(a.compareTo(b));
```

OUTPUT: -17

WHY: Comparing a to b, so the character in a that you are comparing is 'C' and the character in b is 'T'. 'C' has a value of 67 and 'T' has a value of 84. 67 – 84 = -17. Characters C and T have a difference of -17.

```
a = "DataStructure";
b = "DataStructures";
System.out.println(b.compareTo(a));
```

OUTPUT: 1

WHY: Excess character in b compared to a. So the output will yield 1 since that is the difference in number of characters of b to a.

```
a = "DataBase";
b = "Database";
System.out.println(b.compareTo(a));
```

OUTPUT: 32

WHY: Comparing 'b' to 'B' which is 98 to 66. 98 – 66 = 32.

```
a = "Gillead";
b = "Morderna";
System.out.println(b.compareTo(a));

OUTPUT: 6

WHY: Comparing 'M' to 'G' which is 77 to 71. 77 – 71 = 6.



System.out.println(a.compareTo(c));

OUTPUT: -12

WHY: Assuming this program is all one continuous body, then a = Gillead and c = Sorrento. Comparing
'G' to 'S' in this case is 71 – 83 = -12.
```

A.4 - 10 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

Please give 3 examples with detailed analysis to differentiate **Deep Copy** vs. **Shallow Copy** vs. **Mixed Copy**

1. Deep copy is the act of explicitly copying the values of one object to another object. They are not live representations of each other, for example if the original object is altered, the deep copy is not affected.

For example:

```
public class Test{
Static int[] info;

Public static Exam(int[] values){
    Info = new int[values.length];
    For(int i = 0; i < data.length; i++){
        Info[i] = values[i];
}}
Public static void main(String args[]){
    Int num[] = {1,2,3};
    System.out.println(Arrays.toString(info));
    num[0]=0;
    System.out.println(Arrays.toString(info));
}}
```

This program will create a deep copy of the objects, print out the array 1,2,3 and not be affected by the new introduced value 0.

2. Shallow copy creates a new reference to an already existing object. Any changes will be reflected in both the original and the new object. It is performed by the compiler implicitly.

   For example:

```
public class Test{
    static int[] info;
    public static void create(int[] values){
        info = values;
}
```

//continues on page 4…

```
public static void main(String args[]){
int num[] = {1,2,3};
create(num);
System.out.println(Arrays.toString(info));
num[0] = 0;
System.out.println(Arrays.toString(info));
```

This program will create a shallow copy of the objects, print out the array 1,2,3 and then print the shallow copy with 0,1,2,3.

3. Mixed copy, like the name suggests, is a mix of both shallow and deep copy. It performs similarly to a shallow copy however instead of automatically printing the copied data the compiler will need to check if there are any permissions to allow the data in the array to be shared. If the data is allowed to be shared then it will follow through like a deep copy. If not then the changes will remain hidden.

When NetFlix and Disney+ sell their shows via the Internet, should they send **Deep** or **Shallow** or **Mixed** copies to our homes? Please explain in detail.

Mixed copy gives the most control to the people in charge, so Netflix and Disney would definitely want to send mixed copies to homes. There are some users that will be receiving different data depending on what they order or what they want to watch. If there are changes that Netflix or Disney want to make under the radar, they can do so with deep copy and if they want to put out certain changes, they can use shallow. However mixed copies will allow them to pick and choose which data they want to share with certain users. If only deep or shallow was used, then every single user would have the same data shared or same data hidden.

A.5 - 10 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

How is **enum** a special type? Please briefly explain 3 characteristics of **enum.**

1. Enums can contain a constructor. For each constant enum there can be separate constructors executed one by one. We cannot create enum objects though implicitly or explicitly and therefore enum constructors cannot be invoked directly.

2. The way enum works with inheritance is pretty straight forward. Order is very important in enums. Just like in array indexes, we can find each individual enum constant index by using ordinal(). We can use valueOf() to return values of strings if they exist. We can also use values() to return all the values we have present inside the enums. These are all methods that are extended by java.lang.enum. A class can only extend a single parent so we cannot use enum to extend anything else.

3. If there is an abstract method in an enum class then it must always be implemented with each occurrence. Abstract methods and concrete methods can both be in an enum class.

A.6 - 10 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

How does **Shell Sort** work? What are the steps?

Shell sort works in a way where 2 elements within an array are compared who have a decent gap between them in regards to the overall size of the array. We do shell sorting similarly to insertion sorting in the way where we treat the indexes. First we want to make sure our elements are arranged in a table of some organization and then take the size of the array 'n' and divide it by 2. This will be our initial gap, say n=12, our new 'n' will be 6. N is the gap, so we should go down the columns with 6 unit gaps between the elements with insertion sort. This is like taking a very large array and making smaller insertion sorts out of them, in order to make the work more simple and easier to handle. Sometimes if an array index is very large it can be difficult to sort the entire thing one element at a time. With shell sort it cuts out the fat of the process and is much more efficient at tackling larger arrays. After the initial n/2 we would continue to divide by 2 until we reach n/2=1. This is when we can be certain that our shell sort is done correctly and completely.

8 2 9 7 6 5 1 3 4

Show the contents of the array above **each time** an **Insertion Sort** changes it while sorting the array into **ascending order**.

8.	**2.**												
2	**8**	**9**											
2	**8.**	**9**	**7.**										
2	**7.**	**8**	**9**	**6.**									
2	**6.**	**7**	**8**	**9**	**5.**								
2.	**5**	**6**	**7**	**8**	**9**	**1.**							
1	**2**	**5.**	**6**	**7**	**8**	**9**	**3.**						
1	**2**	**3**	**5.**	**6**	**7**	**8**	**9**	**4.**					
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**					**DONE**

A.7 - 20 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

You are hired to create a simple Dictionary application. Your dictionary can take a search key from users then returns value(s) associated with the key.

- Please explain **how** you would implement your dictionary.
- Please state clearly **which data structure(s)** you will use and explain your decision.
- Please explain how you communicate between the data structures and between a data structure and an interface.

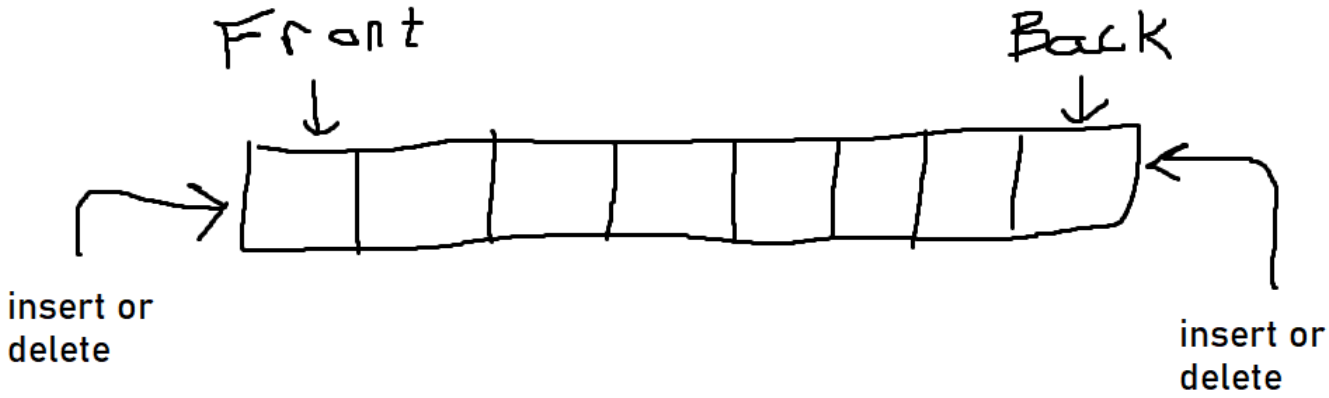
To create a simple Dictionary application, I believe the best course of action to take would be key value pair insertion. This would be the most efficient method. We can use keys and corresponding values to store the data like that of a dictionary. In order to apply this, we will need to use hashmaps so that we can store the data in the aforementioned key value format. Each unique key will have a hash code, and those keys will be stored in a hashtable. Each key will be stored as a string and its value as a list by using hashmap. Data will need to be extracted from one data structure and used in another. In order to combine data structures to build an efficient system we will use hashmap with other data structures list to store multiple values of a specific key. This is done in order to communicate between our data structures. Data will need to be extracted from one data structure and used in another.

PART B – 85 Points

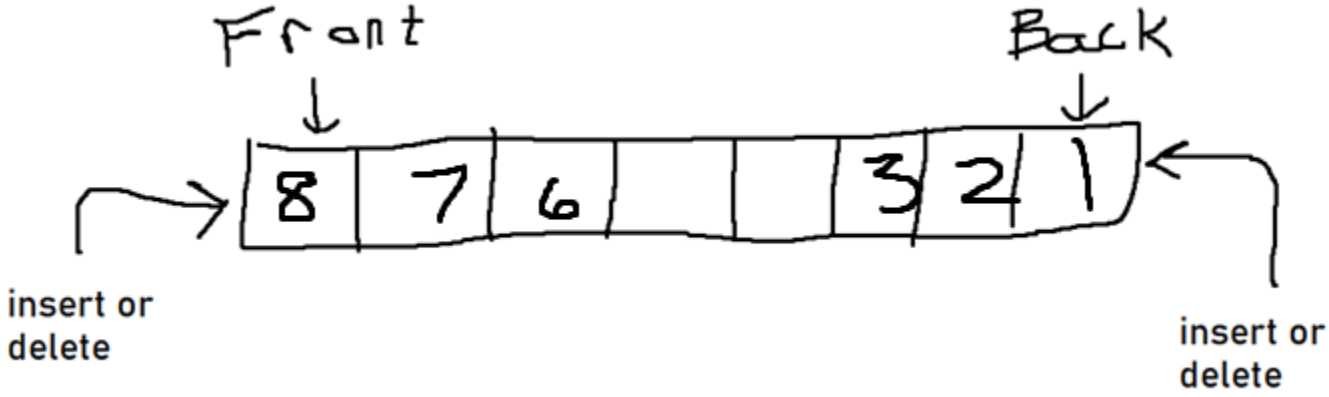
B.1 - 20 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

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

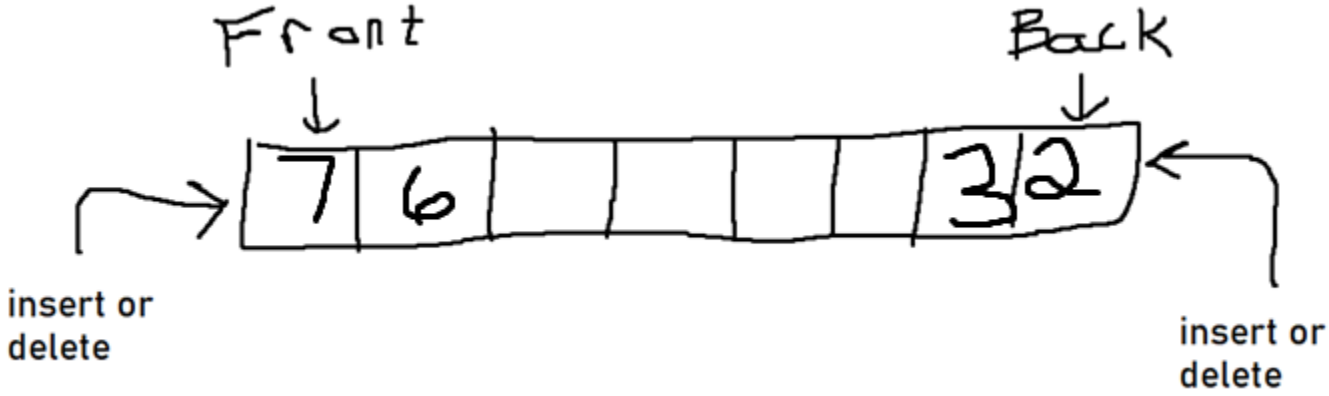
A deque is a queue where you can remove and insert the elements from the front of the back. This insertion and removal of elements can be done at the same time. Diagram below will demonstrate my point:



Usually a deque is implemented using an array and the first element's index is stored in the front. Hence the back is the last element index. If we choose to insert values from the front and the back; say we insert 8, 7, 6, from the front and 1, 2, and 3 from the back simultaneously. It would look like:



And if we were to remove from the front and the back simultaneously again because we can, say we removed 8 and 1, it would look like:



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

```
Deque d = new LinkedList<>();
d.addLast("T");
d.addFirst("e");
d.addFirst("s");
d.addLast("L");
d.add("a");
d.offer(d.remove(d.contains("S")));       // 1
d.addLast(d.remove(d.contains("T")));  // 2
d.push(d.remove("T".toLowerCase())); // 3
d.offer(d.remove());                          // 4
d.offerFirst(d.remove(d.element()));      // 5
```

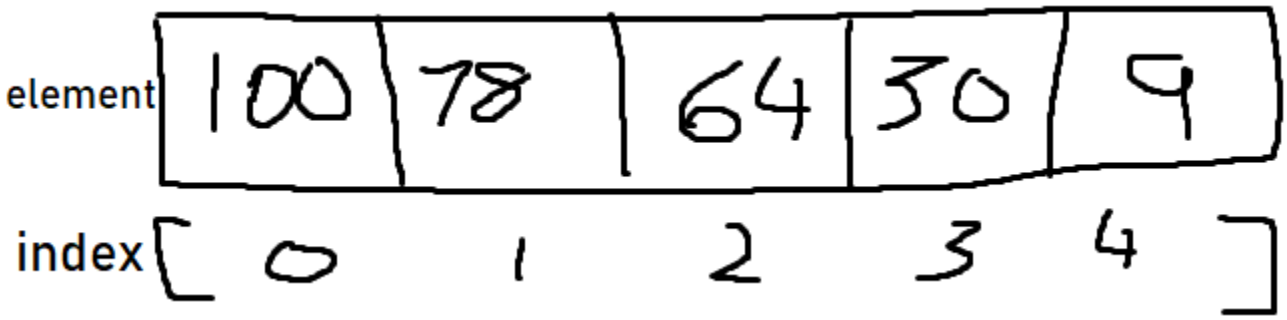
FRONT

1.	**[s, e, T, L, a, false]**
2.	**[s, e, T, L, a, false, false]**
3.	**[false, s, e, T, L, a, false, false]**
4.	**[s, e, T, L, a, false, false, false]**
5.	**[true, e, T, L, a, false, false, false]**

B.2 - 20 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

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

A list is a data type that holds data of the same type. Lists have indexes that start at 0 with elements or values in each index that store data. The following is a diagram of a generic list:



To access the elements, you can use the index number to reference that cell. The size of the element must be a whole integer within the java library.

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

```
LinkedList li = new LinkedList();
for (int i = 8; i >= 0; i--) {
   li.addLast(i);
}
li.remove(li.get(3));
li.add(li.size());
li.set(5, li.size() % 7);                    // 1
li.addFirst(li.get(li.size() - 4));          // 2
li.add(li.indexOf(4));                       // 3
li.addFirst(li.remove(li.indexOf(5)));       // 4
li.addFirst(li.peekLast());                  // 5
System.out.println(li);
```

FRONT

1.	**[8, 7, 6, 4, 3, 2, 1, 0, 8]**
2.	**[2, 8, 7, 6, 4, 3, 2 ,1, 0, 8]**
3.	**[2, 8, 7, 6, 4, 3, 2 ,1, 0, 8, 4]**
4.	**[2, 8, 7, 6, 4, 3, 2 ,1, 0, 8, 4]**
5.	**[4, 2, 8, 7, 6, 4, 3, 2 ,1, 0, 8, 4]**

B.3 - 15 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

Please explain how to import **Google Guava** into a NetBeans Java with Ant

In order to import Google Guava into NetBeans you want to go into the libraries by clicking on the “Tools” tab. From there you want to import the jar for guava manually into your NetBeans application.

Write a statement to create a **Google Guava Multimap** instance which contains student ids as keys and the associated values are student profiles.

And write code to iterate through all entries in this Google Guava multimap and display all the student names in the associated student profiles.

B.4 - 15 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

Given a **deque d** of String elements, please write code to output:

- "*Every other.*" if the elements are organized in these patterns:
  - a, OK, b, OK, c, OK, d
  - a, OK, b, OK, c, OK, d, OK
  - OK, a, OK, b, OK, c, OK, d
  - OK, a, OK, b, OK, c, OK, d, OK
- "*NOT every other.*" if other patterns.

```
public class Deque {
   public static void main(String[] args) {

      Deque<String> d1 = new LinkedList<>();
      Deque<String> d2 = new LinkedList<>();
      d1.add("a");
      d1.add("OK");
      d1.add("b");
      d1.add("OK");
      d1.add("c");
      d1.add("OK");
      d1.add("d");
      System.out.println("d1 is "+ d1);
      if(check(d1)){
         System.out.println("Every other");
      }
      else {
         System.out.println("NOT every other");
      }
      d2.add("OK");
      d2.add("OK");
      d2.add("b");
      d2.add("OK");
      System.out.println("\nd2 is " + d2);
      if(check(d2)){
         System.out.println("Every other");
      }
      else{
         System.out.println("NOT every other");
      }}
   private static boolean check(Deque<String> d) {
      String[] test = {
            "[a, OK, b, OK, c, OK, d]",
            "[a, OK, b, OK, c, OK, d, OK]",
            "[OK, a, OK, b, OK, c, OK, d]",
            "[OK, a, OK, b, OK, c, OK, d, OK]"
      };
      for(String s:test)
         if(s.equals(d.toString()))
            return true;
      return false;
}}
```

B.5 - 15 Points – *Your answer must be in your own words, be in complete sentences, and provide very specific details to earn credit.*

Given 2 Priority Queues containing String objects, please write code to detect if the contents in these data structures are identical. A same string but in uppercase or lowercase or mixed cases can be considered as an identical content. Our program should output meaningful message(s). *Hint: It may take more thinking than doing.*

```
class LanguageChecker implements Check<String>
  {
     public int compare(String firstString, String secondString)
     {
          String string1;
          String string2;
          string1 = firstString;
          string2 = secondString;
          return string2.compareTo(string1);
}}

Public class pqString
{
  public static void main(String[] args)
  {
     PriorityQueue<String> pq1 = new PriorityQueue<>();
     pq1.add("Hello");
     pq1.add("Hola");
     pq1.add("Salam");
     pq1.add("NiHao");

     PriorityQueue<String> pq2 = new PriorityQueue<>();
     pq1.add("NiHao");
     pq1.add("Salam");
     pq1.add("Hola");
     pq1.add("Hello");

     Boolean answer = comparePriorityQueue(pq1, pq2);
     System.out.println("The comparison yields: " + answer);
}
```

PART C – 10 Extra Credit Points

Class **Entry** has 1 property: String **finalExam**. Please write the **hashCode** method for this class using **Joshua Bloch's** recommendation.

And write the **equals()** method to compare 2 objects of this class.