Question 1

Question a

[20 marks]

1 (a) Show how the values below would be sorted by the following algorithms, indicating clearly the swaps involved: [3 marks]

i) Bubble sort
ii) Selection sort
iii) Insertion sort

29 85 32 69 40 12

i) Bubble Sort

[29, 85, 32, 69, 40, 12]

Round 1: [29, 32, 69, 40, 12, 85]
Round 2: [29, 32, 40, 12, 69, 85]
Round 3: [29, 32, 12, 40, 69, 85]
Round 4: [29, 12, 32, 40, 69, 85]
Round 5: [12, 29, 32, 40, 69, 85]

Finally, sorted list: [12, 29, 32, 40, 69, 85]

ii) Selection Sort

[29, 85, 32, 69, 40, 12]

Select min 12, swap with 29: [12, 85, 32, 69, 40, 29]
Select min 29, swap with 85: [12, 29, 32, 69, 40, 85]
Select min 32, no swap:
Select min 40, swap with 69: [12, 29, 32, 40, 69, 85]
Select min 69, no swap
Select min 85, no swap

Finally, sorted list: [12, 29, 32, 40, 69, 85]

iii) Insertion Sort

[29, 85, 32, 69, 40, 12]

insert 29, sorted list: [29, 85, 32, 69, 40, 12]
insert 85, sorted list: [29, 85, 32, 69, 40, 12]
insert 32, sorted list: [29, 32, 85, 69, 40, 12]
insert 69, sorted list: [29, 32, 69, 85, 40, 12]
insert 40, sorted list: [29, 32, 40, 69, 85, 12]
insert 12, sorted list: [12, 29, 32, 40, 69, 85]

Finally, sorted list: [12, 29, 32, 40, 69, 85]

CS210-2023-January

Question b

(b) What output is printed by the following program? Explain your reasoning. [3 marks]

```
public class BitManipulation{
    public static void main(String[] args){
        System.out.println((23|45)^((18&26)<<4));
    }
}
```

The program runs main function first, it will print out the equation `((23|45)^(18&26)<<4)`

Step 1: 23 | 45

$(23)_{10}$	$=(00010111)_2$	
$(45)_{10}$	$=(00101101)_2$	I
	$`(00111111)_2$	$= (63)_{10}$

Step 2: 18 & 26

$(18)_{10}$	$=(00010010)_2$	
$(26)_{10}$	$=(00011010)_2$	&
	$`(00010010)_2$	$= (18)_{10}$

Step 3: 18 << 4

$(00010010)_2$ << 4 = $(100100000)_2$ = $(288)_{10}$

Step 4: 63 ^ 288

$(63)_{10}$	$=(000111111)_2$	
$(288)_{10}$	$=(100100000)_2$	^
	$`(100011111)_2$	$= (287)_{10}$

Therefore, the Java Program outputs 287 when it runs.

Question c

(c) What output is printed by the following program? Explain your reasoning.

```
public class Recursion{

   public static void main(String[] args){
        System.out.println(recursion(26));
   }

   public static int recursion(int x){
       if(x % 9 < 3){
           return 8;
       }else{
           return x % recursion(x - 2) + 1;
       }
   }
}
```

The program runs main function first, it will call the function `recursion(26)`

1) Calling recursion(26).

26 % 9 = 8, 8 > 3 => execute else branch

=> return **26% recursion(26 - 2) + 1**

2) Calling recursion(24).

24 % 9 = 6, 6 > 3 => execute else branch

=> return **24% recursion(24 - 2) + 1**

3) Calling recursion(22).

22 % 9 = 4, 4 > 3 => execute else branch

=> return **22% recursion(22 - 2) + 1**

4) Calling recursion(20).

20 % 9 = 2, 2 < 3 => execute if branch

=> return **8**

5) Calculating recursion(22).

22 % recursion(22 - 2) + 1 = 22 % 8 + 1 = 6 + 1 = 7

=> return **7**

6) Calculating recursion(24).

24 % recursion(24 - 2) + 1 = 24 % 7 + 1 = 3 + 1 = 4

=> return **2**

7) Calculating recursion(26).

26 % recursion(26 - 2) + 1 = 26 % 4 + 1 = 2 + 1 = 3

=> return **3**

Therefore, the Java Program outputs 3 when it runs.

CS210-2023-January

Question d

(d) Show where the values 170, 316 and 273 would be inserted into the following hash table using quadratic probing [3 marks]

0	
1	
2	596
3	
4	345
5	
6	
7	623
8	52
9	713
10	274

Hash Table size = 11.

1) **Insert 170.**
   170 % 11 = 5, 5 is vacant.
   Insert at slot 5.

2) **Insert 316.**
   316 % 11 = 8, 8 is occupied.
   Quaduatic Probing 8-9-2-0
   Insert at slot 0.

3) **Insert 273.**
   273 % 11 = 9 9 s occupied.
   Quaduatic Probing 9-10-3
   Insert at slot 3.

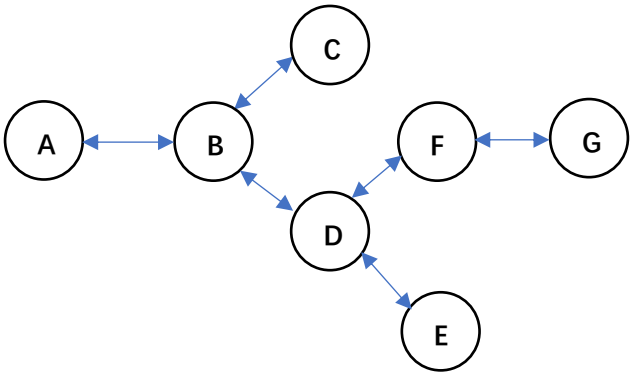
CS210-2023-January

Question e

(e) Draw the directed graph represented by the adjacency matrix below. Show how i) a depth-first and ii) a breadth-first search would traverse the graph starting at A.

	A	B	C	D	E	F	G
A	0	1	0	0	0	0	0
B	1	0	1	1	0	0	0
C	0	1	0	0	0	0	0
D	0	1	0	0	1	1	0
E	0	0	0	1	0	0	0
F	0	0	0	1	0	0	1
G	0	0	0	0	0	1	0

Directed Graph



Depth-First Search		Breadth-First Search	
Events	**Stack**	**Events**	**Queue**
Push A	A	Add A	A
Push B	A B	Remove A	-
Push C	A B C	Add B	B
Pop C	A B	Remove B	-
Push D	A B D	Add C	C
Push F	A B D F	Add D	C D
Push G	A B D F G	Remove C	D
Pop G	A B D F	Remove D	-
Pop F	A B D	Add F	F
Push E	A B D E	Add E	F E
Pop E	A B D	Remove F	E
Pop D	A B	Remove E	-
Pop B	A	Add G	G
Pop A	-	Remove G	-
Traverse the Graph			
Stack		**Queue**	
A B C D F G E		**A B C D F E G**	

CS210-2023-January

Question f

(f) What is the Big O complexity of the Java method below? Explain your reasoning. [3 marks]

```
public void complexity(int n){
     int counter=0;
     for(int i = n; i>500; i--){
         for(int j=6; j<6*n; j=j+n){
             counter++;
         }
     }
}
```

The Big O Complexity of this program is determined by the input **`int n`.**

There are two for loops in this program:

1) Outer Loop `for(int i = n; i > 500; i--)`
   It runs ` n – 500 ` times
2) Inner Loop `for(int j = 6; j < 6*n; j = j + n){`
   It runs ` (6*n - 6) / n ` times

Calculate the Total Iterations:

$$Total\ Iterations = (\mathrm{n} - 500) \times (\frac{6\mathrm{n} - 6}{\mathrm{n}})$$

Simplify the expression, ignore constant terms and lower-order terms:

$$Total\ Iterations = \mathrm{n} \times (\ 6 - \frac{6}{\mathrm{n}}) = (\ 6n - 6)$$

Use Big O notation to the Total Iterations:

$$Big\ O\ Notation = O\ (6n - 6) = \ O\ (n)$$

Therefore, the Big O Complexity of the Java Program is O(n).

CS210-2023-January

Question g

(g) Show how the contents of a priority queue would adjust given the following commands, with highest values given highest priority. [2 marks]

insert 35
insert 19
insert 43
remove
insert 98
remove

remove
insert 69
remove

In this program, highest value was given highest priority, therefore the contents' adjustment would be:

Events	Priority Queue
Insert 35	35
Insert 19	35 19
Insert 43	43 35 19
Remove	35 19
Insert 98	98 35 19
Remove	35 19
Remove	19
Insert 69	69 19
Remove	19

Question 2

[20 marks]

2 Write a Java program which takes in an int *x* and prints out the distance from *x* to the nearest prime number. A prime number is a number which is divisible only by itself and 1 (e.g. 2, 3, 5, 7 etc.) If *x* is a prime, then the output should be 0.

Sample input
24

Sample output
1

```
import java.util.Scanner;

public class Q2 {
   public static void main (String args[]) {
      Scanner sc = new Scanner(System.in);
      int input = sc.nextInt();
      sc.close();
      int distance = 0;
      int upperNum = input;
      int lowerNum = input;
      // If neither upperNum nor lowerNum is Prime,
         distance = distance + 1.
      while(!checkPrime(upperNum) && !checkPrime(lowerNum)) {
         upperNum ++;
         lowerNum --;
         distance ++;
      }
      System.out.println(distance);
   }

   public static boolean checkPrime (int input) {
      if(input <= 1)  return false;

      for(int i = 2; i < input; i++) {
         if (input % i == 0) return false;
      }
      return true;
   }
}
```

CS210-2023-January

Question 3

[20 marks]

3 Write a Java program which uses a Monte Carlo simulation to compute the average number of rolls of a dice that are needed to see all 6 numbers come up (i.e. 1, 2, 3, 4, 5, 6).

```
public class Q3 {
   public static void main (String args[]) {
      int N = 10000000;
      int sumRolls = 0;

      for(int i = 0; i < N; i++) {
         // Index 0 - 5 => Dice Number 1 - 6
         boolean dice[] = {false, false, false,
          false, false, false};
         int count = 0;

         while(!checkDice(dice)) {
            int roll = (int)(Math.random() * 6);
            dice[roll] = true;
            count ++;
         }
         sumRolls = sumRolls + count;
      }

      double averageNumber =  (double)sumRolls /(double) N;
      System.out.printf("%.2f\n", averageNumber);
   }

   // Check if all 6 numbers have come up
   public static boolean checkDice (boolean dice[]) {
      for(boolean number : dice) {
         if (number == false) return false;
      }
      return true;
   }
}
```

Question 4

[20 marks]

4 Write a Java program which takes in an int *n*, followed by a list of *n* English words. The program should sort the list of Strings according to how many unique characters they contain. If two Strings contain the same number of unique characters, then they should be sorted alphabetically. The program should print out the sorted list, with words containing the fewest number of unique characters coming first.

Sample input
tree
paper
car
banana

Sample output
banana
car
tree
paper

```
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Set;
import java.util.HashSet;


public class Q4 {
   public static void main (String args[]) {
      Scanner sc = new Scanner(System.in);
      List<String> list = new ArrayList<String>();

      // Input a list of n English words
      int n = Integer.parseInt(sc.nextLine());
      for (int i = 0; i < n; i++) {
         String inputLine = sc.nextLine();
         list.add(inputLine);
      }
```

```
        // Sort the list
        Collections.sort(list, new Comparator<String>() {
            @Override
            public int compare(String s1, String s2) {
                if(getUniqueChar(s1) != getUniqueChar(s2)) {
                    // 1 - Fewer unique Char first
                    return getUniqueChar(s1) - getUniqueChar(s2);
                }
                else {
                    // 2 - Alphabetically
                    return s1.compareTo(s2);
                }
            }
        });

        // Print the output
        for(String s : list) {
            System.out.println(s);
        }
    }

    //Use HashSet to get the number of Unique Char in a word
    public static int getUniqueChar(String input) {
        Set<Character> s = new HashSet<Character>();
        char[] charArray = input.toCharArray();
        for(int i = 0; i < charArray.length; i++) {
            s.add(charArray[i]);
        }
        int uniqueChar = s.size();
        return uniqueChar;
    }
}
```