a.) String compression

Ans:-

```
import java.util.Scanner;

public class StringCompression {
    public static void compress(String str){
        String result="";
        int count=0;
        for(int i=0;i<str.length();i++){
            if(i==0){
                result=result+str.charAt(0);
                count++;

            }
            if((i!=0) && (str.charAt(i)== str.charAt(i-1))){
                count++;
            }else if((i!=0) && (str.charAt(i)!= str.charAt(i-1))){
                result=result+count+str.charAt(i);
                count=1;
            }
            if(i==str.length()-1){
                result=result+count;
            }
        }
        System.out.println(result);

    }
    public static void main(String[] args) {
        System.out.println("Input: ");
        Scanner sc= new Scanner(System.in);
        String str=sc.next();
        sc.close();
        compress(str);

    }
}
```

b.) Linked List

Ans:-

```
import java.io.*;
class LinkedList {
    Node head;
    class Node {
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
    }
    void printNthFromLast(int N)
    {
```

```
        int len = 0;
        Node temp = head;
        while (temp != null) {
            temp = temp.next;
            len++;
        }
        if (len < N)
            return;

        temp = head;
        for (int i = 1; i < len - N + 1; i++)
            temp = temp.next;

        System.out.println(temp.data);
    }
    public void push(int new_data) {
        Node new_node = new Node(new_data);
        new_node.next = head;
        head = new_node;
    }
    public static void main(String[] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(35);
        llist.printNthFromLast(4);
    }
}
```

c.) Stack minimum-

Ans:-

```
import java.util.*;
class stack {
    Stack<Integer> s;
    Integer minEle;

    // Constructor
    stack() { s = new Stack<Integer>(); }

    // Prints minimum element of MyStack
    void getMin()
    {
        // Get the minimum number in the entire stack
        if (s.isEmpty())
            System.out.println("Stack is empty");

            // variable minEle stores the minimum element
            // in the stack.
        else
            System.out.println("Minimum Element in the "
                    + " stack is: " + minEle);
    }
```

```
    // prints top element of MyStack
    void peek()
    {
        if (s.isEmpty()) {
            System.out.println("Stack is empty ");
            return;
        }

        Integer t = s.peek(); // Top element.

        System.out.print("Top Most Element is: ");

        // If t < minEle means minEle stores
        // value of t.
        if (t < minEle)
            System.out.println(minEle);
        else
            System.out.println(t);
    }

    // Removes the top element from MyStack
    void pop()
    {
        if (s.isEmpty()) {
            System.out.println("Stack is empty");
            return;
        }

        System.out.print("Top Most Element Removed: ");
        Integer t = s.pop();

        // Minimum will change as the minimum element
        // of the stack is being removed.
        if (t < minEle) {
            System.out.println(minEle);
            minEle = 2 * minEle - t;
        }

        else
            System.out.println(t);
    }

    // Insert new number into MyStack
    void push(Integer x)
    {
        if (s.isEmpty()) {
            minEle = x;
            s.push(x);
            System.out.println("Number Inserted: " + x);
            return;
        }

        // If new number is less than original minEle
        if (x < minEle) {
            s.push(2 * x - minEle);
            minEle = x;
```

```
        }

        else
            s.push(x);

        System.out.println("Number Inserted: " + x);
    }
};

// Driver Code
public class MyStack {
    public static void main(String[] args)
    {
        stack s = new stack();

        // Function calls
        s.push(5);
        s.push(2);
        s.getMin();
        s.push(3);
        s.push(1);
        s.getMin();
        s.pop();
        s.getMin();
        s.pop();
        s.peek();
    }
}
```

Explain one real world use case where stack is better used data structure than arrays.

Ans:-

Stacks in Data Structures is a linear type of data structure that follows the LIFO (Last-In-First-Out) principle and allows insertion and deletion operations from one end of the stack data structure, that is top. Implementation of the stack can be done by contiguous memory which is an array, and non-contiguous memory which is a linked list. Stack plays a vital role in many applications. Like backtraking.

d.) Given an array of integers representing the elevation of a roof structure at various positions, each position is separated by a unit length, Write a program to determine the amount of water that will be trapped on the roof after heavy rainfall

ans:-
```
import java.util.Scanner;

public class waterTrapped {
    public static int maxWater(int[] arr, int n)
    {
        int res = 0;


        for (int i = 1; i < n - 1; i++) {
            int left = arr[i];
            for (int j = 0; j < i; j++) {
                left = Math.max(left, arr[j]);
            }
```

```
            int right = arr[i];
            for (int j = i + 1; j < n; j++) {
                right = Math.max(right, arr[j]);
            }
            res += Math.min(left, right) - arr[i];
        }
        return res;
    }
    public static void main(String[] args)
    {
        //int[] arr = { 2 ,1,3,0,1,2,3};
        int m;
        Scanner sc= new Scanner(System.in);
        m=sc.nextInt();
        int[]arr=new int[m];
        for(int i=0;i<m;i++){
            arr[i]=sc.nextInt();
        }


        int n = arr.length;

        System.out.print(maxWater(arr, n));
    }
}
```

e.) You will be given a list coin denominations that you can use to tender change to your customers, find the most optimum way to tender the exact change to your customers , the optimum is when you use the least number of coins.
Ans:-

```
import java.util.*;
public class coins {
    static int count(int coins[], int n, int sum)
    {


        if (sum == 0)
            return 1;


        if (sum < 0)
            return 0;


        if (n <= 0)
            return 0;


        return count(coins, n - 1, sum)
                + count(coins, n, sum - coins[n - 1]);
    }

    // Driver code
    public static void main(String args[])
    {
```

```
        int coins[] = { 1, 2, 5, 8, 10 };
        int n = coins.length;

        System.out.println(count(coins, n, 5));
    }
}
```

f.) What is dot product and cross product? Explain use cases of where dot product is used and cross product is used in graphics environment. Add links to places where you studied this information and get back with the understanding.
Ans:- While both involve multiplying the magnitudes of two vectors, the dot product results in a scalar quantity,which indicates magnitudes but not direction, while the cross product results in a vector,which indicates magnitudes and direction.The cross product is mostly used to determine the vector, which is perpendicular to the plane surface spanned by two vectors, ewhereas the dot product is used to find the angle between two vectors or the length of the vector.
Link:
https://www.khanacademy.org/science/physics/magnetic-forces-and-magnetic-fields/electric-motors/v/dot-vs-cross-product#:~:text=While%20both%20involve%20multiplying%20the,which%20indicates%20magnitude%20and%20direction.

g.) Explain a piece of code that you wrote which you are proud of? If you have not written anycode, please write your favorite subject in engineering studies. We can go deep into that subject.

Ans:- my favourite subject in engineering studies is java.

h.) Random crashes – you are given a source code to test and it randomly crashes and it never crashes in the same place (you have attached a debugger and you find this). Explain what all you would suspect and how would you go about with isolating the cause.

Ans:-

This is how i proceed.

1.Get a crash dump

2.Isolate a set of potential suspicious functionn

3.Tighten up state checking

4.Repeat

I declare that I have done the above work by myself and not worked with anyone or got help from

any individual on the internet.