Assignment Day-13&14

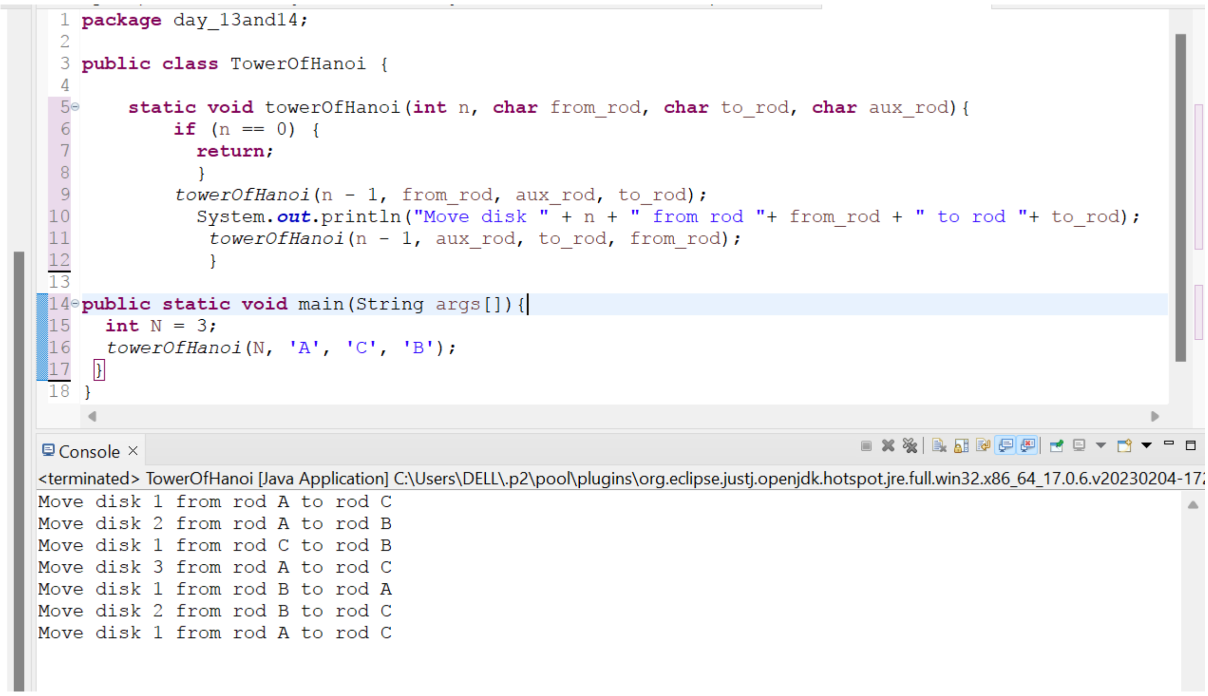
Core Java with DS and Algorithms

Name: Joshnitha Rangolu

Task 1: Tower of Hanoi Solver

Create a program that solves the Tower of Hanoi puzzle for n disks. The solution should use recursion to move disks between three pegs (source, auxiliary, and destination) according to the game's rules. The program should print out each move required to solve the puzzle.

```
package day_13and14;

public class TowerOfHanoi {

static void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod){

if (n == 0) {

return;

}

towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);

System.out.println("Move disk " + n + " from rod "+ from_rod + " to rod "+ to_rod);

towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);

}

public static void main(String args[]){

int N = 3;

towerOfHanoi(N, 'A', 'C', 'B');

}

}
```



```java
package day_13and14;

public class TowerOfHanoi {

    static void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod){
        if (n == 0) {
          return;
          }
        towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
          System.out.println("Move disk " + n + " from rod "+ from_rod + " to rod "+ to_rod);
           towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
           }

public static void main(String args[]){
  int N = 3;
  towerOfHanoi(N, 'A', 'C', 'B');
 }
}
```

```
<terminated> TowerOfHanoi [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-17
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

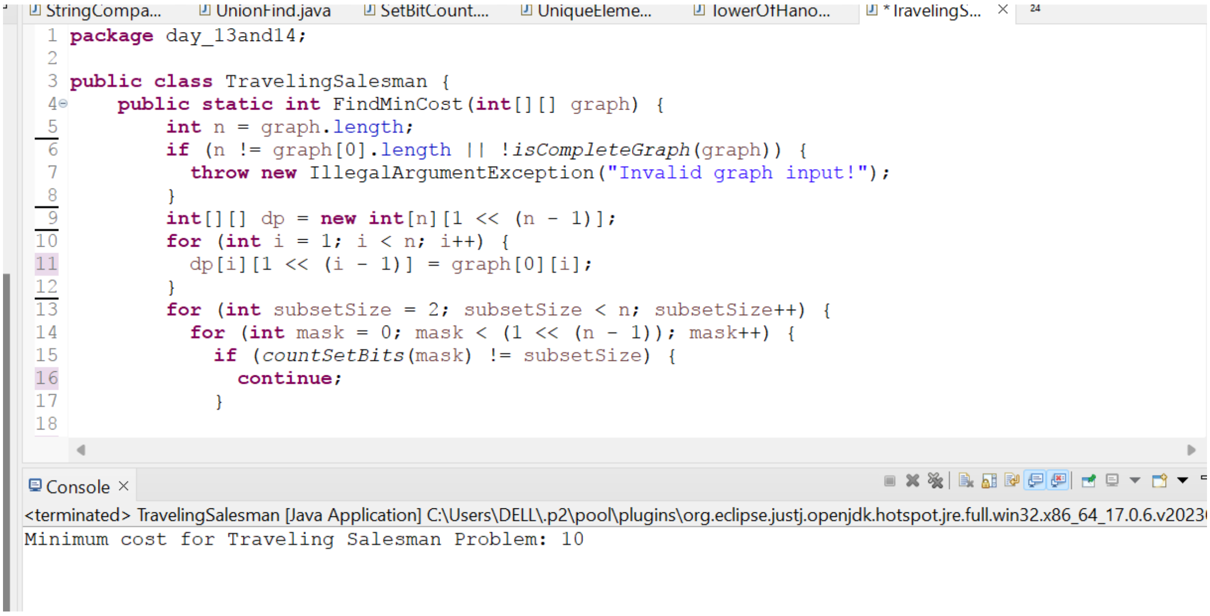
Task 2: Traveling Salesman Problem

Create a function int FindMinCost(int[,] graph) that takes a 2D array representing the graph where graph[i][j] is the cost to travel from city i to city j. The function should return the minimum cost to visit all cities and return to the starting city. Use dynamic programming for this solution.

```java
package day_13and14;

public class TravelingSalesman {

public static int FindMinCost(int[][] graph) {

int n = graph.length;

if (n != graph[0].length || !isCompleteGraph(graph)) {

throw new IllegalArgumentException("Invalid graph input!");

}

int[][] dp = new int[n][1 << (n - 1)];

for (int i = 1; i < n; i++) {

dp[i][1 << (i - 1)] = graph[0][i];

}

for (int subsetSize = 2; subsetSize < n; subsetSize++) {
```

```
for (int mask = 0; mask < (1 << (n - 1)); mask++) {
if (countSetBits(mask) != subsetSize) {
continue;
}
dp[0][mask] = Integer.MAX_VALUE;
for (int j = 1; j < n; j++) {
if ((mask & (1 << (j - 1))) != 0) {
int remainingMask = mask ^ (1 << (j - 1));
dp[0][mask] = Math.min(dp[0][mask], dp[j][remainingMask] + graph[j][0]);
}
}
}
}
return dp[0][(1 << (n - 1)) - 1];
}
private static boolean isCompleteGraph(int[][] graph) {
for (int i = 0; i < graph.length; i++) {
for (int j = 0; j < graph[i].length; j++) {
if (i != j && graph[i][j] == 0) {
return false;
}
}
}
return true;
}
private static int countSetBits(int n) {
int count = 0;
while (n != 0) {
```

```
count += n & 1;
n >>= 1;
}
return count;
}
public static void main(String[] args) {
int[][] graph = {
{0, 10, 15, 20},
{10, 0, 35, 25},
{15, 35, 0, 16},
{20, 25, 16, 0}
};
int minCost = FindMinCost(graph);
System.out.println("Minimum cost for Traveling Salesman Problem: " + minCost);
}
}
```



```
package day_13and14;

public class TravelingSalesman {
    public static int FindMinCost(int[][] graph) {
        int n = graph.length;
        if (n != graph[0].length || !isCompleteGraph(graph)) {
          throw new IllegalArgumentException("Invalid graph input!");
        }
        int[][] dp = new int[n][1 << (n - 1)];
        for (int i = 1; i < n; i++) {
          dp[i][1 << (i - 1)] = graph[0][i];
        }
        for (int subsetSize = 2; subsetSize < n; subsetSize++) {
          for (int mask = 0; mask < (1 << (n - 1)); mask++) {
            if (countSetBits(mask) != subsetSize) {
              continue;
            }
```

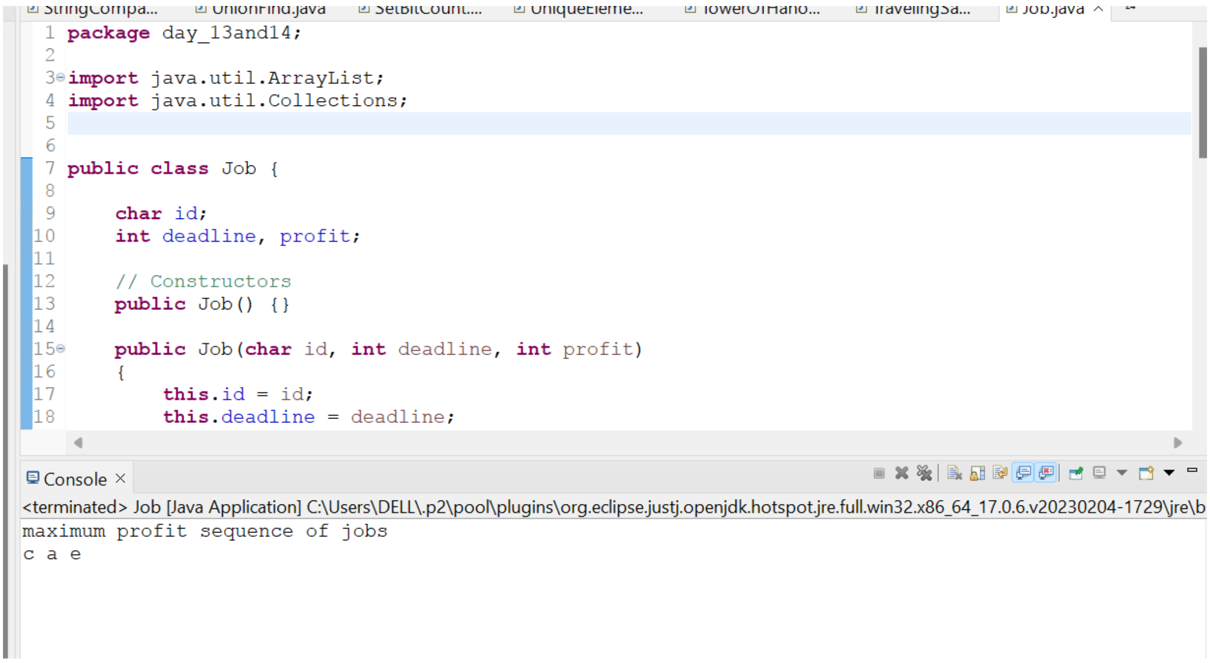
```
<terminated> TravelingSalesman [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v2023
Minimum cost for Traveling Salesman Problem: 10
```

Task 3: Job Sequencing Problem

Define a class Job with properties int Id, int Deadline, and int Profit. Then implement a function List<Job> JobSequencing(List<Job> jobs) that takes a list of jobs and returns the maximum profit sequence of jobs that can be done before the deadlines. Use the greedy method to solve this problem.

```
package day_13and14;
import java.util.ArrayList;
import java.util.Collections;
public class Job {
char id;
int deadline, profit;
// Constructors
public Job() {}
public Job(char id, int deadline, int profit)
{
this.id = id;
this.deadline = deadline;
this.profit = profit;
}
void printJobScheduling(ArrayList<Job> arr, int t)
{
int n = arr.size();
Collections.sort(arr,(a, b) -> b.profit - a.profit);
boolean result[] = new boolean[t];
char job[] = new char[t];
for (int i = 0; i < n; i++) {
for (int j= Math.min(t - 1, arr.get(i).deadline - 1);
j >= 0; j--) {
```

```
if (result[j] == false) {
result[j] = true;
job[j] = arr.get(i).id;
break;
}
}
}
for (char jb : job)
System.out.print(jb + " ");
System.out.println();
}
public static void main(String args[])
{
ArrayList<Job> arr = new ArrayList<Job>();
arr.add(new Job('a', 2, 100));
arr.add(new Job('b', 1, 19));
arr.add(new Job('c', 2, 27));
arr.add(new Job('d', 1, 25));
arr.add(new Job('e', 3, 15));
System.out.println(
"maximum profit sequence of jobs");
Job job = new Job();
job.printJobScheduling(arr, 3);
}
}
```



```java
package day_13and14;

import java.util.ArrayList;
import java.util.Collections;


public class Job {

    char id;
    int deadline, profit;

    // Constructors
    public Job() {}

    public Job(char id, int deadline, int profit)
    {
        this.id = id;
        this.deadline = deadline;
```

```
<terminated> Job [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\b
maximum profit sequence of jobs
c a e
```