FIT1045 Algorithmic Problem Solving – Tutorial 6.

Objectives

The **objectives of this tutorial** are:

- To become familiar with the decomposition of a problem into smaller parts or components.
- To investigate the benefits of decomposition.
- To introduce the basic elements of functions.

Task 1

In this task, we will investigate the *decomposition* of a large problem into smaller problems. This is a useful way of dealing with complex tasks.

Two different mobile phone companies named **A Mobile** and **B Mobile** have decided to merge and become a single company **AB Mobile**. The first task of the IT team is to create a new database without duplicated information.

- Discuss how to break down the task of creating the new database into smaller tasks.
- Organise these tasks in functions. Give them a name, description, and specify input and output.
- How can we use these functions to obtain an algorithm to create the new database?

Task 2

Magic Squares Revisited: In this task, we will use decomposition to write an algorithm to determine if a given table is a magic square. Break into four groups. Each group will be responsible for a function in the following algorithm:

```
function isMagicSquare: algorithm checks if T is a magic square.
Input: T is a n x n table with positive integers entries
          and a number sum, the magic sum of the square.
Output: Algorithm outputs True if T is a magic square and False if T is not a magic square.

if (checkValidInput(T)==False)
{
      print Invalid Square
      output False
}
else
{
      if (checkRows(T, sum) == False)
      {
            output False
      }
      else if (checkCols(T, sum)== False)
      {
            output False
      }
      else if (checkDiagonal1(T, sum)==False)
      {
            output False
```

```
        }
        else if (checkDiagonal2(T, sum)==False)
        {
                output False
        }
        else
        {
                output True
        }
}
```

Note: checkDiagonal1 checks the diagonal from the top left corner to the bottom right corner of T, and checkDiagonal2 checks the diagonal from the bottom left corner to the top right corner of T.

Each group should do the following:

- Provide a header for the function including:
  - the name of the function
  - a brief description of what the function does
  - the input to the function
  - the output of the function
- Write an algorithm for the function. You may create additional functions for the function to use if you wish.

Task 3

Write an algorithm **daysInTheMonth** that takes as input the year and the name of the month and outputs the number of days in that month. You should use functions where it is sensible to do so. Give an algorithm for any function used.

Task 4

A description of the puzzle *Instant insanity* which is given at

`http://techieme.in/graph-theory-applications-the-instant-insanity-puzzle/`

There are four cubes. The faces of each cube are coloured. There are four different colours used and each face is coloured with a single colour. The aim is to stack the four cubes in a tower so that each side of the tower has each colour appearing exactly once.

One algorithmic approach to solving this puzzle is to use brute force, but a solution can be found more quickly using graphs as follows:

1. Construct a graph G with vertices corresponding to the colours.
   Edges correspond to pairs of faces on opposite of a cube and are labelled by the cube number. For example, if the 2nd cube has a red face and a green face on opposite sides of a cube then we create an edge from vertex "red" to vertex "green" and label that edge 2.
2. Find two subgraphs of G with the following constraints:
   - Each of the subgraphs must have all four vertices of the original graph.
   - Each subgraph has exactly four edges of the original graph G and no edges in common with the other subgraph.
   - Each subgraph should have one edge labelled 1, one edge labelled 2, one edge labelled 3 and one edge labelled 4.
   - Each vertex in the subgraph has degree 2.
   - Each subgraph has no loops.
3. A solution can be obtained from these subgraphs: one subgraph gives the front and back face of each cube in the tower and the other subgraph gives the left and right face of each cube in the tower.

Determine what functions you'll need in order to implement this algorithm. Do any of these functions require functions themselves? Write the main algorithm with respect to these functions.

Puzzle of the week

Magic Pond Puzzle: There is a magical pond in Clayton. The pond needs n lotus flowers to cover the whole pond. Since the pond is magical, everyday either one or two lotus blossom grow(s). How many days are needed to cover the whole pond with lotus blossoms.