**Banker’s Algorithm Implementation.**

Abdallah Ahmed Mohamed Awadallah

Mahmoud Fathy Sayed Mohamed

Medhat Ashraf Abdo El-Haddad

**Table of Contents**

1. DESCRIPTION 1
   1. Language Used 1
   2. Data Structures Used 1
2. ASSUMPTIONS 2
3. INPUT/OUTPUT 3

# Description

## Language Used

We have decided on using C++ because of its console application and its given freedom of not having to use strictly either, structured or object- oriented programming.

## Data Structures Used

The following -a bit uncommon- data structures have been used:

1. Vectors
2. Tuples
3. Pairs

Vectors have been used in place of dynamic arrays. Steering away from the usage of pointers was the main objective which facilitates the coding process.

Tuples have been used as an alternative for 2D dynamic arrays, we have a vector of tuples which resembles a link between a group of data linking (String: name of process, and 3 vectors: Allocation, Max, Need) of a certain process, but it is a bit different.

Pairs have been used in order to couple the name of the process that is making a request and the request itself, the request being stored into a vector and the process name is being stored in a string, same is done with the process that is making a release.

# Assumptions

1. As soon as the system grants 5 requests for the process P[i], the process is then terminated and releases all allocated resources.
2. The system’s degree of multiprogramming is limited to 10 concurrent processes.
3. The system is limited to 6 Resource Types.

# 3.0 Input/Output

Input:

1. Number of processes

2. Number of resource types

3. Available resources

4. The ability to show/hide the unsafe requests and releases done and the reasons of denying them.

Output:

1.The complete matrix including the processes names and allocation, max, need of these processes

2.The requests and releases done by the processes and which process did this request/release.

3.(OPTIONAL) the unsafe requests/releases denied by the program and the reasons of denial.