Operating Systems Assignment
Lab dated 25-04-2025

NAME : Aman Choudhary

SAP ID : 590011363

BATCH : AI / ML (B-2)

SUBMITTED TO : Dr. Gaurav Kumar

Q1: Write a shell script that prints the multiplication table of a number entered by the user.

- Input: Number (e.g., 7)
- Output:
  7 x 1 = 7

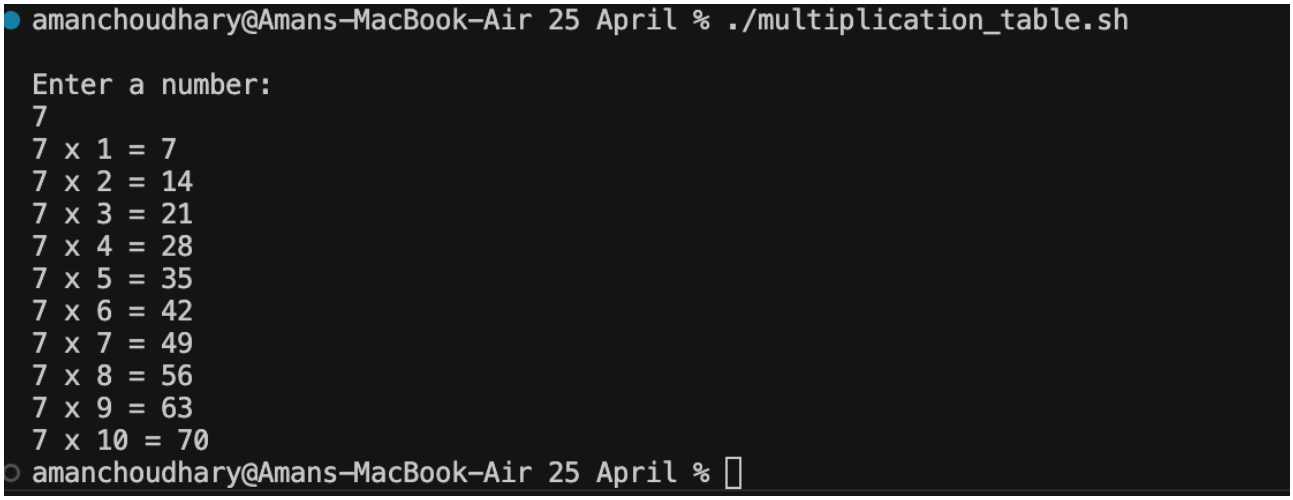
  7 x 2 = 14
  ...
  7 x 10 = 70

SOLUTION:

```
#!/bin/bash

read -p "Enter a number: " num

for i in {1..10}

do
    echo "$num x $i = $((num * i))"
done
```

EXPLANATION:

This script first asks the user for a number. Then, it uses a loop from 1 to 10 to calculate and print the multiplication table for that number in a neat format.



Q2: Write a C program to simulate Deadlock Detection.

SOLUTION:

```c
#include <stdio.h>

int main() {
    int i, j, k, n, m, count = 0;
    int alloc[10][10], max[10][10], need[10][10], avail[10];
    int finish[10] = {0};

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter number of resource types: ");
    scanf("%d", &m);

    printf("Enter Allocation Matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf("%d", &alloc[i][j]);

    printf("Enter Max Matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf("%d", &max[i][j]);

    printf("Enter Available Resources:\n");
    for (j = 0; j < m; j++)
        scanf("%d", &avail[j]);

    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];

    while (count < n) {
        int found = 0;
        for (i = 0; i < n; i++) {
            if (finish[i] == 0) {
                int flag = 1;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]) {
                        flag = 0;
                        break;
                    }
                }
                if (flag == 1) {
```

```
                    for (k = 0; k < m; k++)
                        avail[k] += alloc[i][k];
                    finish[i] = 1;
                    found = 1;
                    count++;
                }
            }
        }
        if (found == 0)
            break;
    }

    int deadlock = 0;
    for (i = 0; i < n; i++) {
        if (finish[i] == 0) {
            deadlock = 1;
            break;
        }
    }

    if (deadlock) {
        printf("\nDeadlock detected. \n");
        for (i = 0; i < n; i++)
            if (finish[i] == 0)
                printf("P%d ", i);
        printf("\n");
    } else {
        printf("\nNo deadlock detected. \n");
    }

    return 0;
}
```

EXPLANATION:

This C program helps detect if a deadlock is happening in a system. It asks for the current resource allocation, the maximum resources each process may need, and what's currently available. Then, it checks if processes can finish one by one. If some processes can't finish even after giving others a chance, those are considered stuck, meaning a deadlock has occurred.

```
amanchoudhary@Amans-MacBook-Air 25 April % gcc deadlock.c -o deadlock

amanchoudhary@Amans-MacBook-Air 25 April % ./deadlock

Enter number of processes: 3
Enter number of resource types: 3
Enter Allocation Matrix:
0 1 0
2 0 0
3 0 3
Enter Max Matrix:
7 5 3
3 2 2
9 0 4
Enter Available Resources:
2 3 0

Deadlock detected.
P0 P1 P2
```