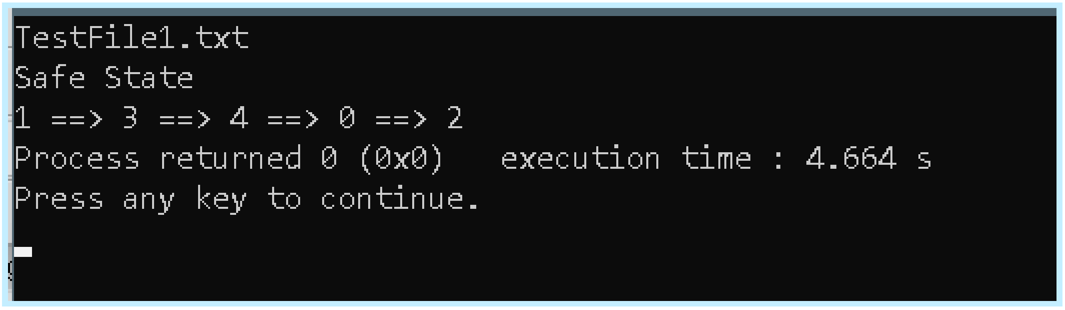
Fatma Ahmed Khalil Ebrahim	*18011227*
Nada Abdo Hanafy Elsadany	*18011917*
Pensée Safwat Mohamed Elessawy	*18010464*
Saeed Mabrouk Saeed El-Shaikh	*18010788*

Code

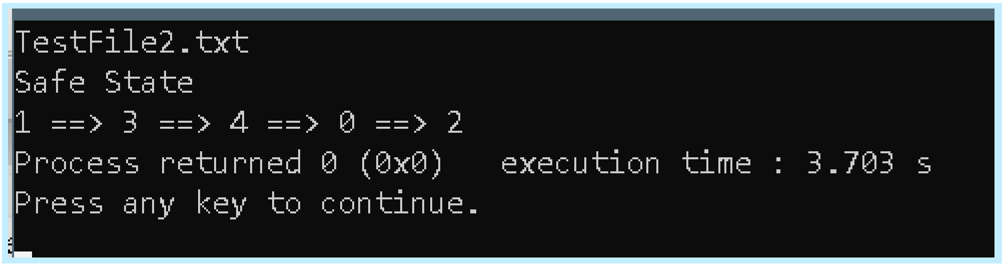
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

//a function fed with file name and array and its job to read the
//numbers from the file and store it in the array
void readmatrix(int rows, int cols, int (*a)[cols], const char* filename)
{
    //pointer to the target file
    FILE *pf;
    pf = fopen (filename, "r");
    for(int i = 0; i < rows; ++i){
        for(int  j = 0; j < cols; ++j){
            //reading single number by single number and store it
            fscanf(pf, "%d", a[i] + j);
        }
    }
    fclose (pf);
    return;
}

int main(void)
{
    char name[15];
    //getting file name
    scanf("%s",name);
    int matrix[11][3];
    readmatrix(11, 3, matrix,name);
    //to store the arrangment of exection of processess
    int exection_stages[5];
    //to show the index of the next exeuted process
    int index =0;
    //allocation matrix
    int allocation[5][3] ;
    //max requerst
    int max[5][3];
    //available resourses
    int available[3];

    //split the read matrix to three matrices
    for(int i = 0; i < 11; ++i)
        for(int j = 0; j < 3; ++j)
            if (i<5)
                allocation[i][j] =matrix[i][j];
            else if (i>=5 && i<10)
                max[i%5][j] =matrix[i][j];

            else
                available[j] =matrix[i][j];


    //reaming resourses for each process = max request - allocation
    int remain[5][3];
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 3; j++)
            remain[i][j] = max[i][j] - allocation[i][j];


    //number of remaining active processes if it down to zero
    // it mean we arrived to safe state
    int p = 5  ;
    bool found;
    while(p != 0){
        found = false ;
        //iterate through all process to find a process we can allocate it
        // some resources if and only if it will end the process
        for(int i = 0 ; i <5; i++ ) {

if(remain[i][0]<=available[0]&&remain[i][1]<=available[1]&&remain[i][2]<=available[2]&&remain[i][
0]!=-1){
                found = true;
                // after ending the process we return the allocated resourses
                //to available resourses
                available[0]+=allocation[i][0];
                available[1]+=allocation[i][1];
                available[2]+=allocation[i][2];
                // we set the remaining resourses for ended process -ve value
                //to easily discover that the process is served before
                remain[i][0]=-1;
                remain[i][1]=-1;
                remain[i][2]=-1;
                p--;
                exection_stages[index]=i;
                index++;
            }
        }

            if(!found) {
                printf("Unsafe State\n");
                return 0;
            }
    }

    if(found)
        printf("Safe State\n");
    for(int i = 0;i <4;i++)
        printf("%d ==> ",exection_stages[i]);
    printf("%d ",exection_stages[4]);
       return 0;
    }
```

Test case 1



```
TestFile1.txt
Safe State
1 ==> 3 ==> 4 ==> 0 ==> 2
Process returned 0 (0x0)   execution time : 4.664 s
Press any key to continue.
```

Test case 2



```
TestFile2.txt
Safe State
1 ==> 3 ==> 4 ==> 0 ==> 2
Process returned 0 (0x0)   execution time : 3.703 s
Press any key to continue.
```

Basic Idea

First: We have used an abstracted function to do the job of reading the input and storing it in a matrix with a size 11*3

Second: we have split the matrix into three matrices

1) 5*3 matrix every row represents the allocated resources for a process
2) 5*3 matrix every row represents the max request for every process
3) 1*3 matrix represents available resources of CPU it can allocate to any process

Third: we have created the reaming resources required for every process by substracting allocated resources from the max request matrix

Fourth: we enter a loop to determine if we can finish all processes and arrive at the safe state or if it falls into a dilemma   and it became an unsafe state in the loop we iterate through the five processes searching for one process we can survive if we find we survive it then store resources in available resources, if we survived all processes it will be a safe state, at any iteration of the bigger loop if we iterated at the 5 processes and we can’t help any process so it will be an unsafe state