```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

int main()
{
    FILE *fp;
    char fileName[MAX], line [MAX];
    int P = 0, R = 0, i, j, z, element;
    char str [10];
    char *pch;
     int K[P], flag = 1, count = 0, safe[P], out [P], o = 0;

    printf("Enter the file name:\t");
    scanf("%s",fileName);

    fp = fopen(fileName,"r");
    while ( fp == NULL )
    {
         system("cls");
         printf("File ( %s ) dose not exist!\n", fileName);
         printf("Enter the file name:\t");
            scanf("%s",fileName);
            fp = fopen(fileName,"r");
     }
    //Reading the first two lines P & R
    fgets(line, sizeof line, fp);
    removeNewLine(line);
    P = atoi (line);

    fgets(line, sizeof line, fp);
    removeNewLine(line);
    R = atoi (line);

    //MAXIMUM array to hold the maximum needs of resources the set of processes P
    int MAXIMUM[P][R];

    //ALLOCATION array to hold the currently allocated resources of the set of processes P
    int ALLOCATION[P][R];

    //AVAILABLE array to hold the currently available resources
    int AVAILABLE[R];

    //NEEDS array to hold the currently needed resources of the set of processes P
    int NEEDS[P][R];


    fgets(line, sizeof line, fp);//to remove the line
    //loop to read the MAXIMUM array
    for (i = 0; i < P; i++)
    {
        j = 0;
        fgets(line, sizeof line, fp);
        removeNewLine(line);

        pch = strtok (line," ");//Split by space

        while(pch != NULL)
        {
            strcpy(str, pch);
            element = atoi (str);
            MAXIMUM[i][j] = element;
            j++;
            pch = strtok (NULL, " ");
        }

    }

    fgets(line, sizeof line, fp);//to remove the line
    //loop to read the ALLOCATION array
    for (i = 0; i < P; i++)
    {
        j = 0;
        fgets(line, sizeof line, fp);
        removeNewLine(line);

        pch = strtok (line," ");//Split by space

        while(pch != NULL)
        {
            strcpy(str, pch);
            element = atoi (str);
            ALLOCATION[i][j] = element;
            j++;
```

```
            pch = strtok (NULL, " ");
        }

    }

    //reading the AVILABLE array
    fgets(line, sizeof line, fp);
    j = 0;
    fgets(line, sizeof line, fp);
    removeNewLine(line);

    pch = strtok (line," ");//Split by space

    while(pch != NULL)
    {
        strcpy(str, pch);
        element = atoi (str);
        AVAILABLE[j] = element;
        j++;
        pch = strtok (NULL, " ");
        }

    fclose(fp);

    //calculate the NEEDS array
    for ( i = 0; i < P; i++)
    {
        for ( j = 0; j < R; j ++)
        {
            NEEDS[i][j] = MAXIMUM[i][j] - ALLOCATION [i][j];
        }

    }



    printf("P = %d \t R = %d\n", P, R);
    printf("\nThe MAXIMUM array:\n");
    for ( i = 0; i < P; i++)
    {
        for ( j = 0; j < R; j ++)
        {
            printf("%d\t", MAXIMUM[i][j]);
        }
        printf("\n");
    }
    printf("\nThe ALLOCATION array:\n");
    for ( i = 0; i < P; i++)
    {
        for ( j = 0; j < R; j ++)
        {
            printf("%d\t", ALLOCATION[i][j]);
        }
        printf("\n");
    }

     printf("\nThe NEEDS array:\n");
    for ( i = 0; i < P; i++)
    {
        for ( j = 0; j < R; j ++)
        {
            printf("%d\t", NEEDS[i][j]);
        }
        printf("\n");
    }

    printf("\nThe AVAILABLE array:\n");
    for ( i = 0; i < R; i++)
    {
        printf("%d\t", AVAILABLE[i]);
    }
    printf("\n");


        //Check if the system is safe
        //Banker algorithm

     for(i = 0;i < P; i++)
         {
          K[i] = 1;
         }

while(flag)   //flag for loop correct continuity
{
```

```
          flag = 0;
            for(i = 0; i < P; i++)
          {
                    int c = 0;
                    for(j = 0; j < R; j++)
                    {
                          if((K[i]== 1)&&(NEEDS[i][j] <= AVAILABLE[j]))
                          {
                               c++;
                               if(c == R)
                                {
                                   for(z = 0; z < R; z++)
                                   {
                                            AVAILABLE[z] += ALLOCATION[i][j];
                                            K[i] = 0;
                                            flag = 1;
                                   }
                                   out[o] = i;
                                   o++;
                                   if(K[i] == 0)
                                   {
                                       i = P;
                                   }
                                }
                          }
                    }
              }
  }

  for(i = 0; i < P; i++)
  {
      if(K[i]== 0)
      {
        count++;
      }
      else
      {
        out[o] = i;
        o++;
      }
  }

  if(count == P)
  {
     printf("\n The system is in safe state");
     printf("\n< ");
     for (i = 0; i < P-1; i++)
     {
         printf("P%d->", out[i]);
     }
     printf("P%d >", out[P-1]);
  }
  else
  {
     printf("\n System are in dead lock");
     printf("\n System is in unsafe state");
  }
     return 0;
}

 //Function to remove \n from a string
 void removeNewLine(char *str)
{
  char *p1 = str, *p2 = str;
  do
    while (*p2 == '\n')
      p2++;
  while (*p1++ = *p2++);
}
```