OS LAB-11 ASSIGNMENT | 4B **Muhammad Danish | cs182019**

CODE SCREENSHOTS:

***function used to calculate need of a process.**

```
#include<stdio.h>
#include <iostream>
#include <queue>
#include <conio.h>
using namespace std;

// p[0],p[1],p[2] = allocation
// p[3],p[4],p[5] = max
// p[6],p[7],p[8] = need
// p[second_last] is for Process Number.
// p[last] is for identifying if process has done completion ?

int p1[]={0,1,0,    7,5,3,  0,0,0   ,1  ,0};
int p2[]={2,0,0,    3,2,2,  0,0,0   ,2  ,0};
int p3[]={3,0,2,    9,0,2,  0,0,0   ,3  ,0};
int p4[]={2,1,1,    2,2,2,  0,0,0   ,4  ,0};
int p5[]={0,0,2,    4,3,3,  0,0,0   ,5  ,0};

int availableResources[]={3,3,2};

queue<int> safeSequence;

void needCalculator(int array[]){
    int j = 3 ;
    int k = 0;
    // calculating need for all the processes
    for(int i=6;i<9;i++)
    {
        array[i]=array[j]-array[k];
        j++;
        k++;
    }

}
```

***function used to print the process allocation, max, need and the available resources.**

```
35  void printer()
36  {
37      int *ptr;
38      ptr=p1;
39      int checker=9;
40      int loopLocker=0; // lock open
41      cout<<" Allocation           Max       Need            Process Num.    Available"<<endl;
42      cout <<"====================|================|====================|===============|=================|"<<endl;
43      int j=0;
44          for(int i=0;i<5;i++)
45          {
46              for(j=0;j<10;j++)
47              {
48                  cout <<*(ptr + j )<< "      ";
49              }
50          if(loopLocker==0)
51          {
52              for(int i=0;i<3;i++)
53              {
54                  cout <<"      "<<availableResources[i];
55              }
56              loopLocker=1;
57          }
58
59          if(j = checker)
60          {
61              if(*(ptr + 9) == 1)
62                  ptr = p2;
63              else if(*(ptr + 9) == 2)
64                  ptr = p3;
65              else if(*(ptr + 9) == 3)
66                  ptr = p4;
67              else if(*(ptr + 9) == 4)
68                  ptr = p5;
69          }
70          cout<<endl;
71      }
72
73  }
```

*Banker's Algorithm

```
int algoRunCounter=0;
void BankersAlgorithm()
{
    int numberOfProcessLeft=5;
    int checker=9;
    int totalProcessCount=5;
    int ProcessExecutedCounter=1;
    bool printOriginalSequence = true;

    while(ProcessExecutedCounter <= totalProcessCount)
    {
        bool SystemSafeCheck = false;
        int *ptr = p1;
        int trueCheck=3;
        for(int i=0;i<5;i++)
        {

            int a =0;
            int allocatorCheck=0; // false
            int j=6;

            // first checking if the process has done completion or not.
            // 0 means it has not done completion.
            if(*(ptr+10) == 0)
            {
                // check if current process need is less or equal to available resources.
                for(j=6;j<9;j++)
                {
                    if(*(ptr + j) <= availableResources[a] )
                        allocatorCheck++;

                    else
                        allocatorCheck--;

                    a++;
                }
                // if the current process is allocate-able than
                if(allocatorCheck == trueCheck && j == checker)
```

```
    }
    // if the current process is allocate-able than
    if(allocatorCheck == trueCheck && j == checker)
    {
        SystemSafeCheck = true;
        if(printOriginalSequence==true)
        {
            cout<<"\n--------- ORIGINAL  SEQUENCE AS PROCESSES ARE ARRANGED ------------"<<endl<<endl;
            printOriginalSequence=false;
        }
        printer();
        cout<<endl;
        cout<<"--------------------------------------------------------------------------------"<<endl;
        cout<<"\nProcess "<<*(ptr+9)<< " Got Executed and set its ALLOCATION FREE."<<endl;
        if(SystemSafeCheck==true)
            cout<<"\n ---> SYSTEM IN IS IN SAFE CONDITION. <-----"<<endl;

        // the process is allocated.
        // now its allocation is set free
        for(int i=0;i<3;i++)
            availableResources[i] =availableResources[i] + *(ptr + i);

        safeSequence.push(*(ptr+9));
        *(ptr + 10)=1; // process has exited(done processing)
        ProcessExecutedCounter++;
        numberOfProcessLeft--;
        cout<<"\nNumber of Processes  Left -- > | "<<numberOfProcessLeft<<" | "<<endl<<endl;

        // after allocation move to next process
        if(*(ptr + 9) == 1)
            ptr = p2;

        else if(*(ptr + 9) == 2 )
            ptr = p3;

        else if(*(ptr + 9) == 3 )
            ptr = p4;

        else if(*(ptr + 9) == 4)
            ptr = p5;

        }
        // if the current process is not allocate-able than
        // than move to next process
        else if (allocatorCheck != trueCheck && j == checker)
        {
            if(*(ptr + 9) == 1)
                ptr = p2;

            else if(*(ptr + 9) == 2 )
                ptr = p3;

            else if(*(ptr + 9) == 3 )
                ptr = p4;

            else if(*(ptr + 9) == 4)
                ptr = p5;
        }
    }
```

```
            // if procees has done completion than move to the next process.
            else
            {
                if(*(ptr + 9) == 1)
                    ptr = p2;

                else if(*(ptr + 9) == 2)
                    ptr = p3;

                else if(*(ptr + 9) == 3)
                    ptr = p4;

                else if(*(ptr + 9) == 4)
                    ptr = p5;
            }
        }

        // if no process is allocatable than exit.
        if(SystemSafeCheck == false)
        {
            cout<<"\nSystem is in Deadlocked State."<<endl;
            cout<<"\nALGORITHM TERMINATED."<<endl;
            exit(0);
        }

        algoRunCounter++;
    }
}
```

*MAIN

```
int main()
{
    //calculating need for every process
    needCalculator(p1);
    needCalculator(p2);
    needCalculator(p3);
    needCalculator(p4);
    needCalculator(p5);

    BankersAlgorithm();

    cout <<endl<<"Safe Sequence Is : ";
    while(!safeSequence.empty())
    {
        cout << "\t --> " << safeSequence.front();
        safeSequence.pop();
    }
    cout <<endl;

    cout <<"\nNumber of Time's the Banker Algorithm has run : "<< algoRunCounter<<endl;

    cout<<"\nPress Any to Exit.."<<endl;
    getch();
    return 0;
}
```

CODE OUTPUT SCREENSHOTS:

```
--------- ORIGINAL  SEQUENCE AS PROCESSES ARE ARRANGED ------------

 Allocation            Max              Need               Process Num.        Available
====================|================|====================|===============|=================|
0      1      0      7      5      3      7      4      3      1             3      3      2
2      0      0      3      2      2      1      2      2      2
3      0      2      9      0      2      6      0      0      3
2      1      1      2      2      2      0      1      1      4
0      0      2      4      3      3      4      3      1      5

--------------------------------------------------------------------------------

Process 2 Got Executed and set its ALLOCATION FREE.

 ---> SYSTEM IN IS IN SAFE CONDITION. <-----

Number of Processes  Left -- > | 4 |

 Allocation            Max              Need               Process Num.        Available
====================|================|====================|===============|=================|
0      1      0      7      5      3      7      4      3      1             5      3      2
2      0      0      3      2      2      1      2      2      2
3      0      2      9      0      2      6      0      0      3
2      1      1      2      2      2      0      1      1      4
0      0      2      4      3      3      4      3      1      5

--------------------------------------------------------------------------------


Process 4 Got Executed and set its ALLOCATION FREE.

 ---> SYSTEM IN IS IN SAFE CONDITION. <-----

Number of Processes  Left -- > | 3 |

 Allocation            Max              Need               Process Num.        Available
====================|================|====================|===============|=================|
0      1      0      7      5      3      7      4      3      1             7      4      3
2      0      0      3      2      2      1      2      2      2
3      0      2      9      0      2      6      0      0      3
2      1      1      2      2      2      0      1      1      4
0      0      2      4      3      3      4      3      1      5

--------------------------------------------------------------------------------

Process 5 Got Executed and set its ALLOCATION FREE.

 ---> SYSTEM IN IS IN SAFE CONDITION. <-----

Number of Processes  Left -- > | 2 |

 Allocation            Max              Need               Process Num.        Available
====================|================|====================|===============|=================|
0      1      0      7      5      3      7      4      3      1             7      4      5
2      0      0      3      2      2      1      2      2      2
3      0      2      9      0      2      6      0      0      3
2      1      1      2      2      2      0      1      1      4
0      0      2      4      3      3      4      3      1      5

--------------------------------------------------------------------------------
```

```
Process 1 Got Executed and set its ALLOCATION FREE.

 ---> SYSTEM IN IS IN SAFE CONDITION. <-----

Number of Processes  Left -- > | 1 |

 Allocation            Max              Need                 Process Num.        Available
====================|================|====================|===============|=================|
0      1      0      7      5      3      7      4      3      1             7      5      5
2      0      0      3      2      2      1      2      2      2
3      0      2      9      0      2      6      0      0      3
2      1      1      2      2      2      0      1      1      4
0      0      2      4      3      3      4      3      1      5

--------------------------------------------------------------------------------

Process 3 Got Executed and set its ALLOCATION FREE.

 ---> SYSTEM IN IS IN SAFE CONDITION. <-----

Number of Processes  Left -- > | 0 |


Safe Sequence Is :       --> 2   --> 4   --> 5   --> 1   --> 3

Number of Time's the Banker Algorithm has run : 2

Press Any to Exit..
```