Banker's Algorithm and Peterson's Solution

Banker's Algorithm

Code

```
#include <stdio.h>

int main() {
   int n, m, i, j, k;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   printf("Enter the number of resources: ");
   scanf("%d", &m);

   int alloc[n][m];
   printf("Enter the allocation matrix:\n");
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
         scanf("%d", &alloc[i][j]);
      }
   }

   int max[n][m];
   printf("Enter the max matrix:\n");
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
         scanf("%d", &max[i][j]);
      }
   }
```

```
int avail[m];
printf("Enter the available resources:\n");
for (j = 0; j < m; j++) {
   scanf("%d", &avail[j]);
}

int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++) {
   f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) {
   for (j = 0; j < m; j++) {
      need[i][j] = max[i][j] - alloc[i][j];
   }
}
int y = 0;
for (k = 0; k < n; k++) {
   for (i = 0; i < n; i++) {
      if (f[i] == 0) {
         int flag = 0;
         for (j = 0; j < m; j++) {
            if (need[i][j] > avail[j]) {
               flag = 1;
               break;
            }
         }
         if (flag == 0) {
            ans[ind++] = i;
            for (y = 0; y < m; y++) {
               avail[y] += alloc[i][y];
```

```
                }
                f[i] = 1;
            }
        }
    }
    int flag = 1;
    for (i = 0; i < n; i++) {
        if (f[i] == 0) {
            flag = 0;
            printf("The following system is not safe");
            break;
        }
    }
    if (flag == 1) {
        printf("Following is the SAFE Sequence:\n");
        for (i = 0; i < n - 1; i++) {
            printf(" P%d ->", ans[i]);
        }
        printf(" P%d\n", ans[n - 1]);
    }
    return 0;
}
```

Output

```
┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ g++ OS.c

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ ./a.out
Enter the number of processes: 4
Enter the number of resources: 3
Enter the allocation matrix:
1 2 3
2 0 1
1 2 3
0 2 1
Enter the max matrix:
4 5 6
6 4 2
2 1 3
1 2 3
Enter the available resources:
4 4 4
Following is the SAFE Sequence:
 P0 -> P1 -> P2 -> P3
```

Peterson's Solution

Code

```
#include<pthread.h>
#include<stdio.h>

void *func1(void *);
void *func2(void *);


int flag[2];
int turn=0;
int global=100;


int main()
{
   pthread_t tid1,tid2;
```

```
    pthread_create(&tid1,NULL,func1,NULL);
    pthread_create(&tid2,NULL,func2,NULL);
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
}



void *func1(void *param)
{
    int i=0;
    while(i<2)
    {
        flag[0]=1;
        turn=1;
        while(flag[1]==1 && turn==1);
        global+=100;
        printf("FT: g: %d",global);
        flag[0]=0;
        i++;
    }
}



void *func2(void *param)
{
    int i=0;
    while(i<2)
    {
        flag[1]=1;
```

```
        turn=0;
        while(flag[0]==1 && turn==0);
        global-=75;
        printf("SP: g: %d",global);
        flag[1]=0;
        i++;
    }
}
```

Output