Deadlock Avoidance using Banker's Algorithm

```
#include <stdio.h>

void main() {
    int n, m, i, j, k, y=0, flag;
    int alloc[100][100], max[100][100],
        avail[100], need[100][100];
    int *f, *ans, ind=0;

    printf("enter no. of processes: ");
    scanf("%d", &n);
    printf("enter no. 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 demand of each process: ");
    for (i=0; i<n; i++) {
        for (j=0; j<m; j++) {
            scanf("%d", &max[i][j]);
        }
    }
    printf("enter available resources: ");
    for (i=0; i<m; i++) {
        scanf("%d", &avail[i]);
    }
    f = (int*) malloc (n * sizeof(int));
    ans = (int*) malloc (n * sizeof(int));
```

```
for (k=0; k<n; k++){
        f[k]=0;
    }
for (i=0; i<n; i++){
    for (j=0; j<m; j++){
        need[i][j]= max[i][j]-alloc[i][j];
}
for (k=0; k<5; k++){
    for (i=0; i<n; i++){
        if (f[i]==0){
            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;
}}}}
flag=1;

for (i=0; i<n; i++){
if (f[i]==0){
    flag=0;
    printf("following system isn't 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", ans[n-1]);
}
}
```

OUTPUT-

```
enter no. of processes in system: 2
enter no. of resource types: 3
enter the allocation matrix:
  4 5
  6 7
  8 9
enter max demand of each process:
  1 2
  2 3
  3 4
enter the available resources: 5 6 7
following is the SAFE sequence
      P0 → P1
```

Deadlock Detection

```
#include <stdio.h>

int main() {
    int i, j, np, nr;
    int *mark, **alloc, **request, *avail, *r, *w;

    printf("enter no. of processes: ");
    scanf("%d", &np);
    printf("enter no. of resources: ");
    scanf("%d", &nr);
    r = (int*)malloc(nr * sizeof(int));
    avail = (int*) malloc(nr * sizeof(int));
    w = (int*) malloc(nr * sizeof(int));
    mark = (int*) malloc(np * sizeof(int));
    request = (int**) malloc(np * sizeof(int*));
    for (i=0; i<np; i++)
        scanf("%d", request[i] = (int*) malloc
                                  (nr * sizeof(int));
                }
    alloc = (int**) malloc(np * sizeof(int*));
    for (i=0; i<np; i++) {
        alloc[i] = (int*) malloc(nr * sizeof(int));
    }
    printf("Total amount of resources: ");
    for (i=0; i<nr; i++)
        scanf("%d", &r[i]);
    printf("enter request matrix: ");
    for (i=0; i<np; i++) {
        for (j=0; j<nr; j++) {
            scanf("%d", &request[i][j]);
    }
```

```
printf("enter allocation matrix:");
for (i=0; i<np; i++) {
    for (j=0; j<nr; j++){
        scanf("%d", &alloc[i][j]);
}
for (j=0; j<nr; j++){
    avail[j] = r[j];
    for (i=0; i<np; i++){
        avail[j] -= alloc[i][j];
}}
for (i=0; i<np; i++) {
    int count=0;
    for (j=0; j<nr; j++){
        if (alloc[i][j]==0)
            count++;
        else
            break;
    }
    if (count==nr)
        mark[i]=1;
}
for (j=0; j<nr; j++)
    w[j] = avail[j];
for (i=0; i<np; i++) {
    int canbeprocessed=0;
    if (mark[i]!=1){
        for (j=0; j<nr; j++) {
            if (request[i][j]<=w[j])
                canbeprocessed=1;
            else {
                canbeprocessed=0;
                break;
}}
```

```
        if (canbeprocessed){
            mark[i]=1;
            for(j=0;j<nr;j++){
                w[j]+=alloc[i][j];
        }}}

    int deadlock=0;
    for(i=0;i<np;i++){
        if(mark[i]!=1)
            deadlock=1;
        if(deadlock)
            printf("\n deadlock detected");
        else
            printf("\n no deadlock possible");
    }
```

OUTPUT

```
enter no. of processes: 2
enter no. of resources: 3
total amount of resources: 8 9 10
enter request matrix: 2 3 5
                      5 4 6
enter allocation matrix: 9 8 7
                         6 5 4
deadlock detected.
```

27/7