```
/* Banker's Algorithm */

#include <stdio.h>
#include <conio.h>

typedef enum {false, true} boolean;

typedef struct State {
	int resource[10];
	int available[10];
	int claim[20][10];
	int alloc[20][10];
} state;

state S;

int np, nr, seq[20], sc=0;

void copyState(state *oldst, state *newst) {

	int i, j;

	for(i=0; i<nr; i++) {
		(*newst).resource[i] = (*oldst).resource[i];
		(*newst).available[i] = (*oldst).available[i];
		for(j=0; j<np; j++) {
			(*newst).claim[j][i] = (*oldst).claim[j][i];
			(*newst).alloc[j][i] = (*oldst).alloc[j][i];
		}
	}
}

boolean safe(state s) {

	int currentavail[10], rest[20], i, j;

	boolean possible = true, found = false;

	for(i=0; i<np; i++) rest[i] = 1;

	for(i=0; i<nr; i++) currentavail[i]=s.available[i];

	while(possible) {

		for(i=0; i<np&&found==false; i++) {
			if (rest[i]!=1) continue;
			for(j=0; j<nr; j++)
				if(s.claim[i][j]-s.alloc[i][j]<=currentavail[j])
					found = true;
				else {
					found = false;
					break;
				}
		}
		if(found) {

			for(j=0; j<nr; j++)
				currentavail[j] = currentavail[j] + s.alloc[i-1][j];

			rest[i-1] = 0;
			found = false;
			seq[sc++] = i-1;
		}

		else
			possible = false;
	}

	for(i=1; i<np; i++)
		rest[0]+=rest[i];

	return (rest[0]==0);

}

void allocRes(int p, int r, int n) {

	int i, j;
	state newstate;

	if(S.alloc[p][r]+n>S.claim[p][r]) {
		printf("\n\tCannot allocate resource. Total request is greater than claim.");
		return;
	}
```

```
	else if(n>S.available[r]) {
		printf("\n\tNot enough free resources available. Process is suspended.");
		return;
	}

	else {
		copyState(&S, &newstate);
		newstate.alloc[p][r] = newstate.alloc[p][r] + n;
		newstate.available[r] = newstate.available[r] - n;
	}

	if(safe(newstate)) {
		printf("\n\tSystem is safe after allocation. Resource is allocated.");
		printf("\n\tSafe sequence: ");
		for(i=0; i<np; i++)
			printf("%d ",seq[i]);

		copyState(&newstate, &S);
	}

	else
		printf("\n\tSystem is unsafe after allocation. Resource is not allocated.");

}

void main() {

	int i, j, n;

	char ch='Y';

	while(ch=='y'||ch=='Y') {

		clrscr();

		printf("Enter system state:\n");

		printf("\n\tNumber of processes: ");
		scanf("%d",&np);

		printf("\n\tNumber of resources: ");
		scanf("%d",&nr);

		printf("\n\tResource vector: ");
		for(i=0; i<nr; i++)
			scanf("%d",&S.resource[i]);

		printf("\n\tAvailable vector: ");
		for(i=0; i<nr; i++)
			scanf("%d",&S.available[i]);

		printf("\n\tClaim matrix:\n");
		for(i=0; i<np; i++) {
			printf("\t\tP%d: ",i);
			for(j=0; j<nr; j++)
				scanf("%d",&S.claim[i][j]);
		}

		printf("\n\tAllocation matrix:\n");
		for(i=0; i<np; i++) {
			printf("\t\tP%d: ",i);
			for(j=0; j<nr; j++)
				scanf("%d",&S.alloc[i][j]);
		}

		printf("\n\nEnter process, resource and requirement: ");
		scanf("%d %d %d",&i,&j,&n);

		allocRes(i, j, n);

		getch();

		printf("\n\nDo you want to continue? (Y/N) : ");
		ch=getch();
	}

}
```

```
Enter system state:

        Number of processes: 4

        Number of resources: 3

        Resource vector: 9 3 6

        Available vector: 0 1 1

        Claim matrix:
                P0: 3 2 2
                P1: 6 1 3
                P2: 3 1 4
                P3: 4 2 2

        Allocation matrix:
                P0: 1 0 0
                P1: 6 1 2
                P2: 2 1 1
                P3: 0 0 2


Enter process, resource and requirement: 1 2 1

        System is safe after allocation. Resource is allocated.
        Safe sequence: 1 0 2 3

Do you want to continue? (Y/N) : Y



Enter system state:

        Number of processes: 4

        Number of resources: 3

        Resource vector: 9 3 6

        Available vector: 1 1 2

        Claim matrix:
                P0: 3 2 2
                P1: 6 1 3
                P2: 3 1 4
                P3: 4 2 2

        Allocation matrix:
                P0: 1 0 0
                P1: 5 1 1
                P2: 2 1 1
                P3: 0 0 2


Enter process, resource and requirement: 0 0 1

       System is unsafe after allocation. Resource is not allocated.

Do you want to continue? (Y/N) : N
```