14.Banker’s Algorithm

```
#include<stdio.h>
int main() {
 int p, c, count = 0, i, j, alc[5][3], max[5][3], need[5][3], safe[5], available[3], done[5], terminate = 0;
 printf("Enter the number of process and resources");
 scanf("%d %d", & p, & c);
 printf("enter allocation of resource of all process %dx%d matrix", p, c);
 for (i = 0; i < p; i++) {
  for (j = 0; j < c; j++) {
   scanf("%d", & alc[i][j]);
  }
 }
 printf("enter the max resource process required %dx%d matrix", p, c);
 for (i = 0; i < p; i++) {
  for (j = 0; j < c; j++) {
   scanf("%d", & max[i][j]);
  }
 }
 printf("enter the  available resource");
 for (i = 0; i < c; i++)
  scanf("%d", & available[i]);

 printf("\n need resources matrix are\n");
 for (i = 0; i < p; i++) {
  for (j = 0; j < c; j++) {
   need[i][j] = max[i][j] - alc[i][j];
   printf("%d\t", need[i][j]);
  }
  printf("\n");
 }
```

```
for (i = 0; i < p; i++) {
  done[i] = 0;
}
while (count < p) {
  for (i = 0; i < p; i++) {
    if (done[i] == 0) {
      for (j = 0; j < c; j++) {
        if (need[i][j] > available[j])
          break;
      }

      if (j == c) {
        safe[count] = i;
        done[i] = 1;
        for (j = 0; j < c; j++) {
          available[j] += alc[i][j];
        }
        count++;
        terminate = 0;
      } else {
        terminate++;
      }
    }
  }
  if (terminate == (p - 1)) {
    printf("safe sequence does not exist");
    break;
  }

}
```

```
  if (terminate != (p - 1)) {
    printf("\n available resource after completion\n");
    for (i = 0; i < c; i++) {
      printf("%d\t", available[i]);
    }
    printf("\n safe sequence are\n");
    for (i = 0; i < p; i++) {
      printf("p%d\t", safe[i]);
    }
  }
  return 0;
}
```