```
#include <stdio.h>

struct file {
   int all[10];
   int max[10];
   int need[10];
   int flag;
};

void findSafeSequence(struct file f[10], int n, int r, int avail[10]) {
   int fl;
   int i, j, k, p, b, g, cnt = 0;
   int seq[10];

   for (i = 0; i < n; i++) {
      for (j = 0; j < r; j++) {
         f[i].need[j] = f[i].max[j] - f[i].all[j];
         if (f[i].need[j] < 0)
            f[i].need[j] = 0;
      }
   }

   while (cnt != n) {
      g = 0;
      for (j = 0; j < n; j++) {
         if (f[j].flag == 0) {
            b = 0;
            for (p = 0; p < r; p++) {
               if (avail[p] >= f[j].need[p])
                  b = b + 1;
               else
```

```
                b = b - 1;
            }
            if (b == r) {
                seq[fl++] = j;
                f[j].flag = 1;
                for (k = 0; k < r; k++)
                    avail[k] = avail[k] + f[j].all[k];
                cnt = cnt + 1;
                g = 1;
            }
        }
    }
    if (g == 0) {
        printf("\n REQUEST NOT GRANTED -- DEADLOCK OCCURRED");
        printf("\n SYSTEM IS IN UNSAFE STATE");
        return;
    }
  }

  printf("\nSYSTEM IS IN SAFE STATE");
  printf("\nThe Safe Sequence is -- (");
  for (i = 0; i < fl; i++)
    printf("P%d ", seq[i]);
  printf(")");
}

int main() {
  struct file f[10];
  int fl;
  int i, j, n, r, g, cnt = 0;
  int avail[10], seq[10],new[10];
```

```
printf("Enter number of processes -- ");
scanf("%d", &n);
printf("Enter number of resources -- ");
scanf("%d", &r);
for (i = 0; i < n; i++) {
   printf("Enter details for P%d", i);
   printf("\nEnter allocation\t -- \t");
   for (j = 0; j < r; j++)
      scanf("%d", &f[i].all[j]);
   printf("Enter Max\t\t -- \t");
   for (j = 0; j < r; j++)
      scanf("%d", &f[i].max[j]);
   f[i].flag = 0;
}

printf("\nEnter Available Resources\t -- \t");
for (i = 0; i < r; i++){
   scanf("%d", &avail[i]);
   new[i]=avail[i];
}

findSafeSequence(f, n, r, avail);

while (1) {
   printf("\nEnter New Request Details -- ");
   printf("\nEnter pid \t -- \t");
   scanf("%d", &g);

   printf("Enter Request for Resources \t -- \t");
   for (i = 0; i < r; i++) {
```

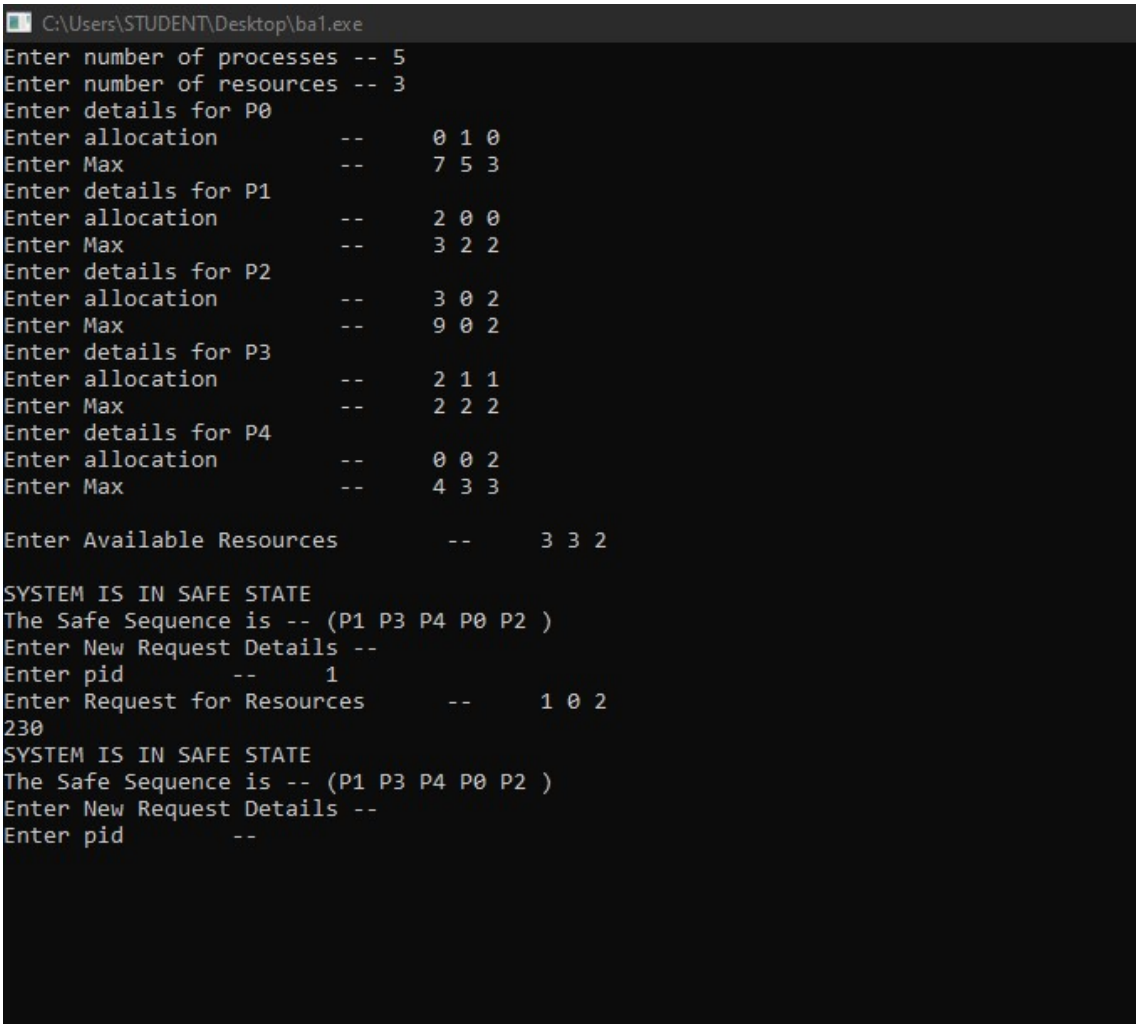
```
            scanf("%d", &fl);
            f[g].all[i] += fl;
            new[i] = new[i] - fl;
        }

        cnt = 0;
        for (i = 0; i < n; i++)
            f[i].flag = 0;


        findSafeSequence(f, n, r, new);
    }

    return 0;
}
```

OUTPUT:


C:\Users\STUDENT\Desktop\ba1.exe
Enter number of processes -- 5
Enter number of resources -- 3
Enter details for P0
Enter allocation -- 0 1 0
Enter Max -- 7 5 3
Enter details for P1
Enter allocation -- 2 0 0
Enter Max -- 3 2 2
Enter details for P2
Enter allocation -- 3 0 2
Enter Max -- 9 0 2
Enter details for P3
Enter allocation -- 2 1 1
Enter Max -- 2 2 2
Enter details for P4
Enter allocation -- 0 0 2
Enter Max -- 4 3 3
Enter Available Resources -- 3 3 2
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2)
Enter New Request Details --
Enter pid -- 1
Enter Request for Resources -- 1 0 2
230
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2)
Enter New Request Details --
Enter pid --