Name : Omkar Arun Shinde PRN No. : 122B1B258

Assignment No. 06

Problem Statement : Write a program to implement Banker's Algorithm for deadlock avoidance.

Code :

```
#include <stdio.h>

#include <stdlib.h>

int n;

int m;

int Count = 0;

int Allocation[10][10];

int Max[10][10];

int Need[10][10];

int Available[10];

int Work[10];

int Request[10];

int Finish[10];

int Sequence[10];

int Arr[10];


int Relation(int x[], int y[]) {

   for (int i = 0; i < m; i++) {

      if (x[i] > y[i]) {

         return 0;

      }

   }

   return 1;

}
```

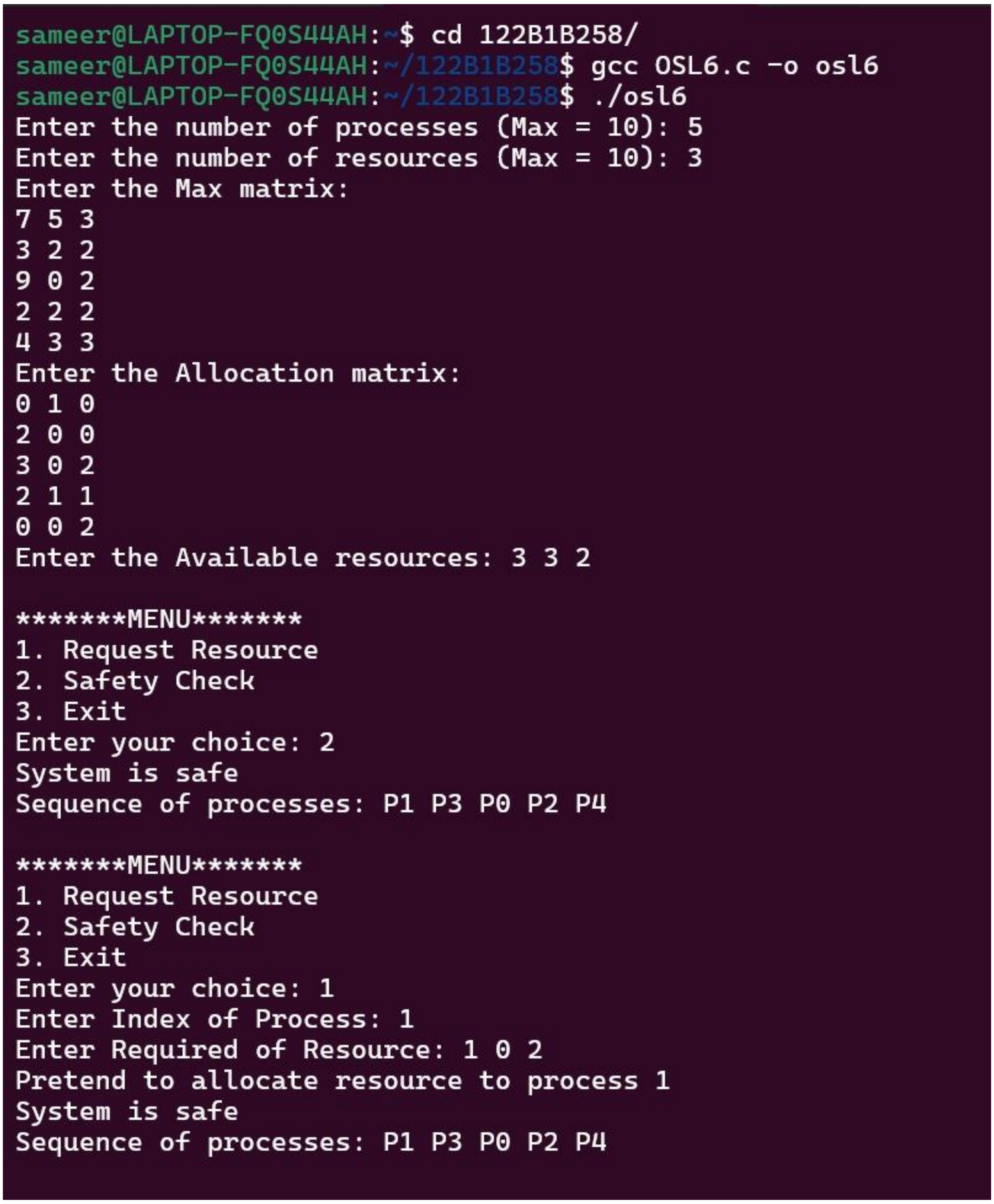
```
int Safety() {
    int flag1 = 0;
    int flag2 = 0;
    int i;
    for (i = 0; i < m; i++) {
        Work[i] = Available[i];
    }
    for (i = 0; i < n; i++) {
        Finish[i] = 0;
    }
    Count = 0;
    while (flag1 == 0) {
        flag2 = 0;
        for (i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                Arr[j] = Need[i][j];
            }
            if ((Finish[i] == 0) && (Relation(Arr, Work) == 1)) {
                flag2 = 1;
                break;
            }
        }
        if (flag2 == 1) {
            Finish[i] = 1;
            Sequence[Count] = i;
            Count++;
            for (int j = 0; j < m; j++) {
                Work[j] = Work[j] + Allocation[i][j];
            }
        } else {
            flag1 = 1;
        }
    }
    for (int i = 0; i < n; i++) {
        if (Finish[i] == 0) {
            printf("System is unsafe\n");
            return 0;
        }
    }
    printf("System is safe\n");
    printf("Sequence of processes: ");
    for (int i = 0; i < n; i++) {
        printf("P%d ", Sequence[i]);
    }
    printf("\n");
    return 1;
}
void Resource() {
    int i, j;
    printf("Enter Index of Process: ");
    scanf("%d", &i);
    printf("Enter Required of Resource: ");
    for (j = 0; j < m; j++) {
        scanf("%d", &Request[j]);
    }
    if (Relation(Request, Need[i])) {
        if (Relation(Request, Available)) {
            printf("Pretend to allocate resource to process %d\n", i);
            for (j = 0; j < m; j++) {
                Available[j] = Available[j] - Request[j];
                Allocation[i][j] = Allocation[i][j] + Request[j];
                Need[i][j] = Need[i][j] - Request[j];
            }
            Safety();
        } else {
            printf("Process must wait...\n");
        }
    } else {
        printf("Resources can't be allocated.\n");
    }
}

int main() {
    int ch = 0;
    printf("Enter the number of processes (Max = 10): ");
    scanf("%d", &n);
    printf("Enter the number of resources (Max = 10): ");
    scanf("%d", &m);
    printf("Enter the Max matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &Max[i][j]);
        }
```

```
}
printf("Enter the Allocation matrix:\n");
for (int i = 0; i < n; i++) {
   for (int j = 0; j < m; j++) {
      scanf("%d", &Allocation[i][j]);
   }
}
for (int i = 0; i < n; i++) {
   for (int j = 0; j < m; j++) {
      Need[i][j] = Max[i][j] - Allocation[i][j];
   }
}
printf("Enter the Available resources: ");
for (int i = 0; i < m; i++) {
   scanf("%d", &Available[i]);
}
do {
   printf("\n*******MENU*******\n");
   printf("1. Request Resource\n");
   printf("2. Safety Check\n");
   printf("3. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &ch);
   switch (ch)
   {
   case 1:
      Resource();
      Break;
```

```
        case 2:
            Safety();
            break;
        case 3:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice\n");
            break;
        }
    } while(ch != 3);
    return 0;
}
```

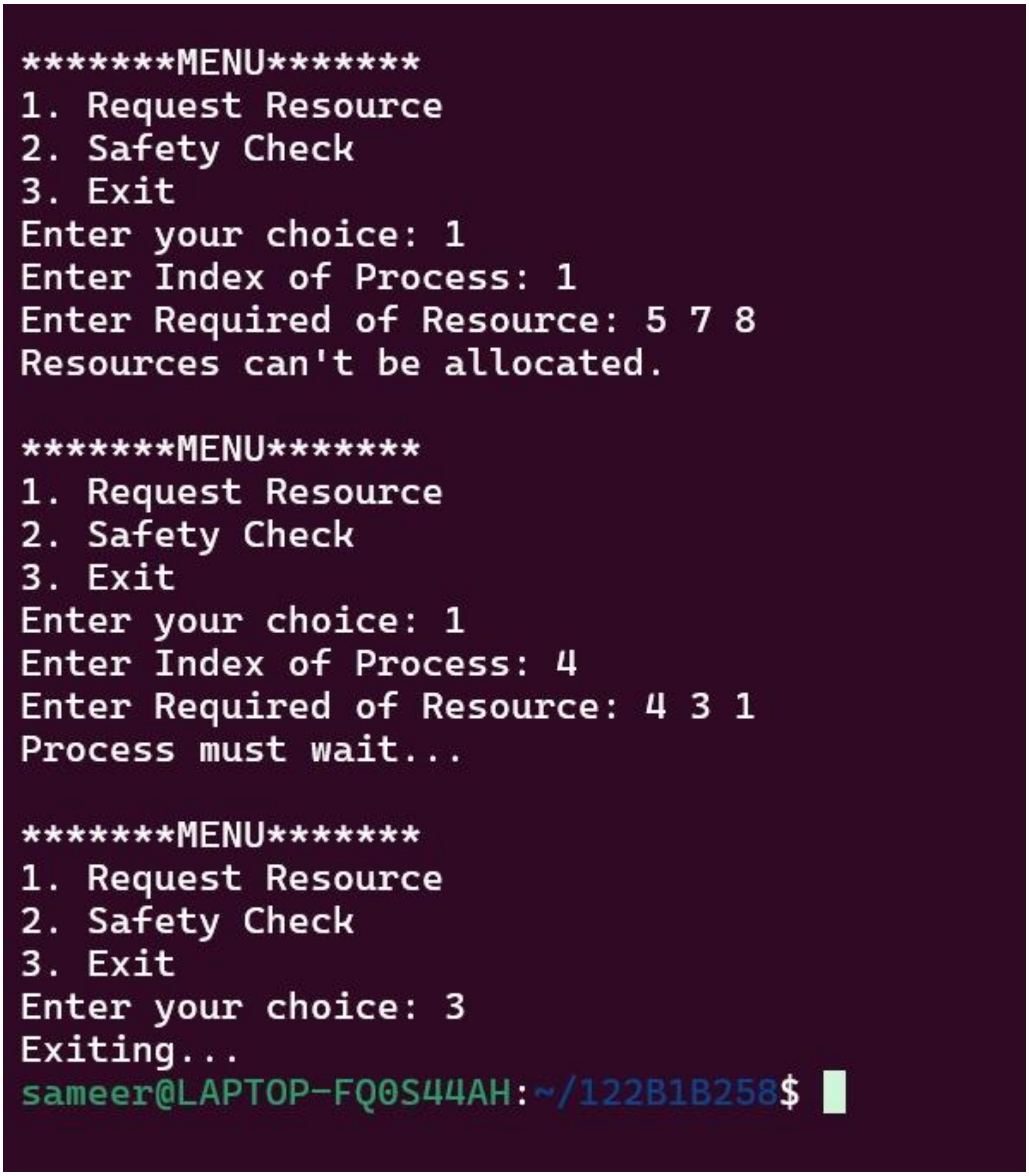
Output :



```
sameer@LAPTOP-FQ0S44AH:~$ cd 122B1B258/
sameer@LAPTOP-FQ0S44AH:~/122B1B258$ gcc OSL6.c -o osl6
sameer@LAPTOP-FQ0S44AH:~/122B1B258$ ./osl6
Enter the number of processes (Max = 10): 5
Enter the number of resources (Max = 10): 3
Enter the Max matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the Available resources: 3 3 2

*******MENU*******
1. Request Resource
2. Safety Check
3. Exit
Enter your choice: 2
System is safe
Sequence of processes: P1 P3 P0 P2 P4

*******MENU*******
1. Request Resource
2. Safety Check
3. Exit
Enter your choice: 1
Enter Index of Process: 1
Enter Required of Resource: 1 0 2
Pretend to allocate resource to process 1
System is safe
Sequence of processes: P1 P3 P0 P2 P4
```



```
*******MENU*******
1. Request Resource
2. Safety Check
3. Exit
Enter your choice: 1
Enter Index of Process: 1
Enter Required of Resource: 5 7 8
Resources can't be allocated.

*******MENU*******
1. Request Resource
2. Safety Check
3. Exit
Enter your choice: 1
Enter Index of Process: 4
Enter Required of Resource: 4 3 1
Process must wait...

*******MENU*******
1. Request Resource
2. Safety Check
3. Exit
Enter your choice: 3
Exiting...
sameer@LAPTOP-FQ0S44AH:~/122B1B258$
```