# Experiment 6
# Banker's Algorithm

**Done By: Rohit Karunakaran**
**Roll No: 58**
**Date Of Submission: 10-08-2021**

**Program Code:**

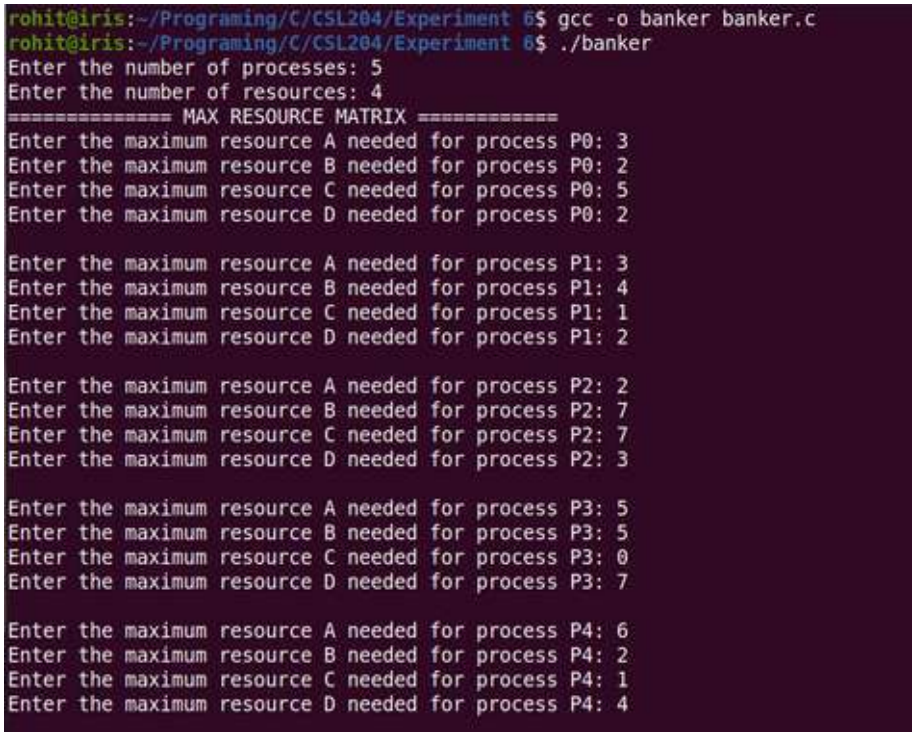
```
#include<stdio.h>
#include<stdlib.h>
#include<stddef.h>



int** get_need_mat(int**, int**, int, int);
int** get_mat(int, int, char[]);
int* get_avail(int);
void delete_mat(int **,int row);
void print_sequence(int*,int);
void print_matrix(int**, int, int);

int* get_safe_sequence(int**,int**, int*,int,int);
int  compare_need_alloc(int*, int*,int);
void run_proc(int*,int*,int);

void
main(){
    int p;
    int r;
    printf("Enter the number of processes: ");
    scanf("%d",&p);

    printf("Enter the number of resources: ");
    scanf("%d",&r);

    printf("============== MAX RESOURCE MATRIX ============\n");
    int ** max_mat = get_mat(p,r,"Enter the max resource %d for process %c: ");

    printf("============== RESOURCE ALLOCATION MATRIX ============\n");
    int ** alloc_mat = get_mat(p,r,"Enter the allocated resource %d for process
%c: ");

    printf("\n\nMAX RESOURCE MATRIX\n");
    print_matrix(max_mat,p,r);

    printf("\nRESOURCE ALLOCATION MATRIX\n");
    print_matrix(alloc_mat,p,r);

    printf("\n");
    int *avail_vec = get_avail(r);

    int *safe_seq = get_safe_sequence(max_mat,alloc_mat,avail_vec,p,r);

    if (safe_seq != NULL){
        printf("Safe sequence is: ");
        print_sequence(safe_seq,p);
        free(safe_seq);
```

```
    }
    else{
        printf("It is in a dead lock state.\n");
    }
    delete_mat(max_mat,p);
    delete_mat(alloc_mat,p);
}

int*
get_safe_sequence(int ** max, int** alloc, int* avail,int procs, int res){
    int r = 0; // keeps track of number of process completed
    int* safe_seq = (int*) malloc(sizeof(int)*procs);
    int** need = get_need_mat(max, alloc, procs, res);
    printf("\n\nNEED MATRIX\n");
    print_matrix(need, procs,res);
    int deadlock = 1; // Deadlock flag, if in a single iteration through the
process list, no process is run the system is in deadlock

    int comp_vec[procs]; // keeps track of completed process to avoid O(n)
searching later on
    for(int i=0;i<procs;i++){
        comp_vec[i] = 0;
    }

    if (need ==NULL) {
        printf("ERROR in creating need Matrix");
        return NULL;
    }

    while(r<procs){ //while there are process remaining
        deadlock = 1;
        for(int i = 0; i < procs; i++){ // Loop over all the processes
            if ( compare_need_alloc(need[i],avail,res) && !comp_vec[i] ){
//Takes O(n) time
                // Process can be run since there is enough resource
                run_proc(max[i], avail,res);
                comp_vec[i] = 1;
                deadlock = 0;
                //printf("Running Process %c\n",i+'A');
                r++;
                safe_seq[r-1] = i;
            }
        }
        if (deadlock){
            return NULL;
        }
    }
    return safe_seq;
}

void run_proc(int* max, int* avail, int res){
    for (int i=0; i<res; i++){
        avail[i]+=max[i];
    }
}
int
compare_need_alloc(int* need, int* avail,int res){
    for(int i=0; i<res; i++){
        if (need[i]>avail[i])
            return 0;
```

```
    }
    return 1;
}


/* UTILITY FUNCTIONS */

int**
get_need_mat(int** max, int** alloc, int proc, int res){
    int** need = (int**)malloc(sizeof(int*)*proc);
    for (int p = 0; p<proc; p++){
        need[p] = (int*) malloc(sizeof(int)*res);
    }

    for (int i=0;i<proc;i++)
        for(int j = 0; j< proc; j++){
            int c = max[i][j]-alloc[i][j];
            if (c<0){
                delete_mat(need,proc);
                return NULL; // Need Cant be less than 0 if it is then it is an
error
            }
            need[i][j] = c;
        }
    return need;
}


int **
get_mat(int procs, int res, char mod_string[]){ //Returns an procs x res matrix
with the max demand of each process
    int i;
    int** max_mat = (int**) malloc(sizeof(int*)*procs); //row initilation
    for (i=0;i<procs; i++)
        max_mat[i] = (int*) malloc(sizeof(int)*res); //initilize all the columns
    int j; // control variable for resources

    for (i=0;i<procs; i++){
        for (j=0;j<res;j++){
            printf(mod_string, j,i+'A');
            scanf("%d",&max_mat[i][j]);
        }
    }
    return max_mat;
}

int*
get_avail(int res){
    int * avail_vec = (int*) malloc(sizeof(int)*res);
    for (int i =0 ; i<res;i++){
        printf("Enter availablity of resource: R%d ",i);
        scanf("%d",avail_vec+i);
    }
    return avail_vec;
}

void
delete_mat(int** mat, int row){
    for (int i=0;i< row; i++){
        free(mat[i]);
```

```
    }
    free(mat);
}

void
print_sequence(int* seq, int proc){
    if (seq!=NULL){
        for (int i = 0; i<proc; i++){
            printf("%c ",seq[i]+'A');
        }
        printf("\n");
    }
}

void print_matrix(int** mat,int row,int col){
    for(int i=0;i<row; i++){
        printf("%c: ",i+'A');
        for(int j=0;j<col;j++){
            printf("%d ",mat[i][j]);
        }
        printf("\n");
    }
}
```

## Screenshots:



```
============== RESOURCE ALLOCATION MATRIX ============
Enter the allocated resource A for process P0: 1
Enter the allocated resource B for process P0: 0
Enter the allocated resource C for process P0: 2
Enter the allocated resource D for process P0: 2

Enter the allocated resource A for process P1: 0
Enter the allocated resource B for process P1: 2
Enter the allocated resource C for process P1: 1
Enter the allocated resource D for process P1: 2

Enter the allocated resource A for process P2: 2
Enter the allocated resource B for process P2: 4
Enter the allocated resource C for process P2: 5
Enter the allocated resource D for process P2: 0

Enter the allocated resource A for process P3: 3
Enter the allocated resource B for process P3: 0
Enter the allocated resource C for process P3: 0
Enter the allocated resource D for process P3: 0

Enter the allocated resource A for process P4: 4
Enter the allocated resource B for process P4: 2
Enter the allocated resource C for process P4: 1
Enter the allocated resource D for process P4: 3
```

```
MAX RESOURCE MATRIX
P0: 3 2 5 2
P1: 3 4 1 2
P2: 2 7 7 3
P3: 5 5 0 7
P4: 6 2 1 4

RESOURCE ALLOCATION MATRIX
P0: 1 0 2 2
P1: 0 2 1 2
P2: 2 4 5 0
P3: 3 0 0 0
P4: 4 2 1 3

Enter availablity of resource A: 3
Enter availablity of resource B: 0
Enter availablity of resource C: 0
Enter availablity of resource D: 1


NEED MATRIX
P0: 2 2 3 0
P1: 3 2 0 0
P2: 0 3 2 3
P3: 2 5 0 7
P4: 2 0 0 1
Safe sequence is: P4 P1 P2 P3 P0
rohit@iris:~/Programing/C/CSL204/Experiment 6$
```

```
rohit@iris:~/Programing/C/CSL204/Experiment 6$ ./banker
Enter the number of processes: 5
Enter the number of resources: 3
============= MAX RESOURCE MATRIX ============
Enter the maximum resource A needed for process P0: 7
Enter the maximum resource B needed for process P0: 5
Enter the maximum resource C needed for process P0: 3

Enter the maximum resource A needed for process P1: 4
Enter the maximum resource B needed for process P1: 2
Enter the maximum resource C needed for process P1: 2

Enter the maximum resource A needed for process P2: 9
Enter the maximum resource B needed for process P2: 0
Enter the maximum resource C needed for process P2: 2

Enter the maximum resource A needed for process P3: 2
Enter the maximum resource B needed for process P3: 5
Enter the maximum resource C needed for process P3: 2

Enter the maximum resource A needed for process P4: 4
Enter the maximum resource B needed for process P4: 3
Enter the maximum resource C needed for process P4: 3
```

```
============= RESOURCE ALLOCATION MATRIX ============
Enter the allocated resource A for process P0: 2
Enter the allocated resource B for process P0: 0
Enter the allocated resource C for process P0: 0

Enter the allocated resource A for process P1: 2
Enter the allocated resource B for process P1: 0
Enter the allocated resource C for process P1: 0

Enter the allocated resource A for process P2: 0
Enter the allocated resource B for process P2: 0
Enter the allocated resource C for process P2: 2

Enter the allocated resource A for process P3: 2
Enter the allocated resource B for process P3: 1
Enter the allocated resource C for process P3: 1

Enter the allocated resource A for process P4: 0
Enter the allocated resource B for process P4: 0
Enter the allocated resource C for process P4: 2
```

```
MAX RESOURCE MATRIX
P0: 7 5 3
P1: 4 2 2
P2: 9 0 2
P3: 2 5 2
P4: 4 3 3

RESOURCE ALLOCATION MATRIX
P0: 2 0 0
P1: 2 0 0
P2: 0 0 2
P3: 2 1 1
P4: 0 0 2

Enter availablity of resource A: 1
Enter availablity of resource B: 3
Enter availablity of resource C: 1


NEED MATRIX
P0: 5 5 3
P1: 2 2 2
P2: 9 0 0
P3: 0 4 1
P4: 4 3 1
It is in an unsafe state.
rohit@iris:~/Programing/C/CSL204/Experiment 6$
```

```
rohit@iris:~/Programing/C/CSL204/Experiment 6$ ./banker
Enter the number of processes: 5
Enter the number of resources: 4
============== MAX RESOURCE MATRIX ============
Enter the maximum resource A needed for process P0: 0
Enter the maximum resource B needed for process P0: 0
Enter the maximum resource C needed for process P0: 1
Enter the maximum resource D needed for process P0: 2

Enter the maximum resource A needed for process P1: 1
Enter the maximum resource B needed for process P1: 7
Enter the maximum resource C needed for process P1: 5
Enter the maximum resource D needed for process P1: 0

Enter the maximum resource A needed for process P2: 2
Enter the maximum resource B needed for process P2: 3
Enter the maximum resource C needed for process P2: 5
Enter the maximum resource D needed for process P2: 6

Enter the maximum resource A needed for process P3: 0
Enter the maximum resource B needed for process P3: 6
Enter the maximum resource C needed for process P3: 5
Enter the maximum resource D needed for process P3: 2

Enter the maximum resource A needed for process P4: 0
Enter the maximum resource B needed for process P4: 6
Enter the maximum resource C needed for process P4: 5
Enter the maximum resource D needed for process P4: 6
```

```
============== RESOURCE ALLOCATION MATRIX =============
Enter the allocated resource A for process P0: 0
Enter the allocated resource B for process P0: 0
Enter the allocated resource C for process P0: 1
Enter the allocated resource D for process P0: 2

Enter the allocated resource A for process P1: 1
Enter the allocated resource B for process P1: 0
Enter the allocated resource C for process P1: 0
Enter the allocated resource D for process P1: 0

Enter the allocated resource A for process P2: 1
Enter the allocated resource B for process P2: 3
Enter the allocated resource C for process P2: 5
Enter the allocated resource D for process P2: 4

Enter the allocated resource A for process P3: 0
Enter the allocated resource B for process P3: 6
Enter the allocated resource C for process P3: 3
Enter the allocated resource D for process P3: 2

Enter the allocated resource A for process P4: 0
Enter the allocated resource B for process P4: 0
Enter the allocated resource C for process P4: 1
Enter the allocated resource D for process P4: 4
```

```
MAX RESOURCE MATRIX
P0: 0 0 1 2
P1: 1 7 5 0
P2: 2 3 5 6
P3: 0 6 5 2
P4: 0 6 5 6

RESOURCE ALLOCATION MATRIX
P0: 0 0 1 2
P1: 1 0 0 0
P2: 1 3 5 4
P3: 0 6 3 2
P4: 0 0 1 4

Enter availablity of resource A: 1
Enter availablity of resource B: 5
Enter availablity of resource C: 2
Enter availablity of resource D: 0


NEED MATRIX
P0: 0 0 0 0
P1: 0 7 5 0
P2: 1 0 0 2
P3: 0 0 2 0
P4: 0 6 4 2
Safe sequence is: P0 P2 P3 P4 P1
rohit@iris:~/Programing/C/CSL204/Experiment 6$
```