UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 7 – Implementation of Banker's Algorithm (Deadlock Avoidance) and Deadlock Detection

Date: 25/04/2022

Aim:

i) To develop a C program to implement Banker's algorithm for deadlock avoidance with multiple instances of resource types.

ii) To develop a C program to implement an algorithm for deadlock detection with multiple instances of resource types and display the processes involved in deadlock.

Algorithm:

i) Banker's Algorithm for Deadlock Avoidance:

1. Start
2. Read the following:
    1. No. of processes, *n.*
    2. No. of resources, *m.*
    3. Total number of instances of each resource.
    4. Maximum requirement of each resource.
    5. Allocated instances of each resource.
    6. Available number of instances of each resource.
3. Calculate the number of resources needed by each process by subtracting the allocated number of resources from the maximum number of resources.
4. Display the given details in a table.
5. To run the safety algorithm and check the system state:
    1. Let the vectors representing the work and finish status be vectors of lengths *m* and *n* respectively. Initialize:
        1. The work vector with the values of the available resources.
        2. The finish vector with 0 for each entry.
    2. Find a process such that its finish value is still 0 (i.e., the process has not been completed) and its need vector is less than the work vector.
    3. If such a process is found, add the resources allocated for the process to the work vector and set the finish states of the process to 1. Add the process to the safety sequence before repeating step 5.b.
    4. If the finish status of every process has been changed to 1, then the system is in a safe state. Print the safety sequence.
    5. If one or more of the processes could not be completed, then display the processes for which the needed resources could not be allocated.
6. To put in a resource request, do the following:

1. Get the process id that requests the resource and the request vector for the number of resources requested as input from the user.
2. If the number of resources requested is lesser than the resources needed as well as the number of resources available, then:
    1. Update the available, needed and allocated vector resources for that process.
    2. Run the safety algorithm and print the safety sequence if it exists.
    3. If a safe sequence is obtained, grant the resources to the process by updating the system state. If not, inform the user that the resource request cannot be granted.
7. Stop

ii) Algorithm for Deadlock Detection:

1. Start
2. Read the following:
    1. No. of processes, *n.*
    2. No. of resources, *m.*
    3. Total number of instances of each resource.
    4. Maximum requirement of each resource.
    5. Allocated instances of each resource.
    6. Available number of instances of each resource.
3. Calculate the number of resources needed by each process by subtracting the allocated number of resources from the maximum number of resources.
4. Display the given details in a table.
5. To run the safety algorithm and check the system state:
    1. Let the vectors representing the work and finish status be vectors of lengths *m* and *n* respectively. Initialize:
        1. The work vector with the values of the available resources.
        2. The finish vector with 0 for each entry.
    2. Find a process such that its finish value is still 0 (i.e., the process has not been completed) and its need vector is less than the work vector.
    3. If such a process is found, add the resources allocated for the process to the work vector and set the finish states of the process to 1. Add the process to the safety sequence before repeating step 5.b.
    4. If the finish status of every process has been changed to 1, then the system is in a safe state. Print the safety sequence.
    5. If one or more of the processes could not be completed, then display the incomplete processes that cause a potential deadlock.
6. Stop

Name: Krithika Swaminathan
Roll No.: 205001057

Programs:

1) Banker's Algorithm for Deadlock Avoidance:

Code:

```
//Implementation of Banker's Algorithm for Deadlock Avoidance

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define P_CAP 10
#define R_CAP 5

void safety();

int main() {
	printf("\n__BANKER'S ALGORITHM__\n\n");

	int n, m, t=0;
	printf("Enter no. of processes: ");
	scanf("%d",&n);
	printf("Enter no. of resources: ");
	scanf("%d",&m);

	char pid[n][3], rid[m];
	int alloc[n][m], max[n][m], need[n][m], avail[n][m], total[m], available[m], work[m],
finish[n], safety[n], reqid, req[m];

	printf("\nEnter details of these resources:-\n\n");
	for (int j=0; j<m; j++) {
		printf("Resource name: ");
		scanf(" %c",&rid[j]);
		printf("\tTotal no. of instances: ");
		scanf("%d",&total[j]);
		}

	printf("\n__MENU__\n1. Read data\n2. Print data\n3. Check system state\n4. Resource
request\n5. Exit\n");

	int choice = 1;
	do {
		printf("\nEnter choice: ");
		scanf("%d",&choice);

		switch (choice) {
			case 1: {
				printf("\nEnter no. of available instances of each resource:-\n\n");
```

```
        for (int i=0; i<m; i++) {
                printf("Resource %c: ",rid[i]);
                scanf("%d",&avail[0][i]);
                work[i] = avail[0][i];
                available[i] = avail[0][i];
                }

        printf("\nEnter no. of processes: ");
        scanf("%d",&n);

        printf("Enter the details for each process:-\n\n");
        for (int i=0; i<n; i++) {
                printf("Process name: ");
                scanf("%s",pid[i]);
                printf("\tResources allocated: ");
                for (int j=0; j<m; j++) {
                        scanf("%d",&alloc[i][j]);
                        }
                printf("\tMaximum resources needed: ");
                for (int j=0; j<m; j++) {
                        scanf("%d",&max[i][j]);
                        }
                for (int j=0; j<m; j++) {
                        need[i][j] = max[i][j] - alloc[i][j];
                        }

                finish[i] = 0;
                }

        break;
        }

case 2: {
        //print in table form
        printf("\nPID\tAllocated\tMaximum\t\tNeed\t\tAvailable\n");
        printf("\t");
        for (int col=1; col<=4; col++) {
                for (int j=0; j<m; j++)
                        printf("%c ",rid[j]);
                printf("\t\t");
                }
        printf("\n");

        for (int i=0; i<n; i++) {
                printf("%s\t",pid[i]);

                for (int j=0; j<m; j++)
                        printf("%d ",alloc[i][j]);
                printf("\t\t");
```

```
                        for (int j=0; j<m; j++)
                                printf("%d ",max[i][j]);
                        printf("\t\t");

                        for (int j=0; j<m; j++)
                                printf("%d ",need[i][j]);
                        printf("\t\t");

                        if (i==0) {
                                for (int j=0; j<m; j++)
                                        printf("%d ",available[j]);
                                }

                        printf("\n");
                        }
                break;
                }

        case 4: {
                //resource request
                printf("Enter pid no. to request: P");
                scanf("%d",&reqid);

                printf("Resources requested: ");
                for (int j=0; j<m; j++)
                        scanf("%d",&req[j]);

                for (int j=0; j<m; j++)
                        work[j] = available[j];

                int status = 1;
                for (int j=0; j<m; j++)
                        if (req[j] > need[reqid][j] || req[j] > work[j])
                                status = 0;
                if (status == 1) {
                        for (int j=0; j<m; j++) {
                                work[j] = work[j] - req[j];
                                alloc[reqid][j] += req[j];
                                need[reqid][j] -= req[j];
                                avail[reqid][j] = work[j];
                                }
                        }
                else {
                        printf("The request cannot be satisfied with the current
resources.\n");

                        break;
                        }
```

```
printf("Process %s:\n",pid[reqid]);
printf("Available: ");
for (int j=0; j<m; j++)
        printf("%d ",avail[reqid][j]);
printf("\n");

printf("Allocation: ");
for (int j=0; j<m; j++)
        printf("%d ",alloc[reqid][j]);
printf("\n");

printf("Need: ");
for (int j=0; j<m; j++)
        printf("%d ",need[reqid][j]);
printf("\n");

//print table
printf("\nPID\tAllocated\tMaximum\t\tNeed\t\tAvailable\n");
printf("\t");
for (int col=1; col<=4; col++) {
        for (int j=0; j<m; j++)
                printf("%c ",rid[j]);
        printf("\t\t");
        }
printf("\n");
for (int i=0; i<n; i++) {
        printf("%s\t",pid[i]);

        for (int j=0; j<m; j++)
                printf("%d ",alloc[i][j]);
        printf("\t\t");

        for (int j=0; j<m; j++)
                printf("%d ",max[i][j]);
        printf("\t\t");

        for (int j=0; j<m; j++)
                printf("%d ",need[i][j]);
        printf("\t\t");

        if (i==0) {
                for (int j=0; j<m; j++)
                        printf("%d ",work[j]);
                }

        printf("\n");
        }
}
```

```
case 3: {
        //display current status
        if (choice==3) {
                for (int i=0; i<n; i++) {
                        for (int j=0; j<m; j++) {
                                need[i][j] = max[i][j] - alloc[i][j];
                                }
                        }
                }

        for (int k=0; k<n; k++)
                finish[k] = 0;

        printf("\nChecking the status of the set of processes:- \n");

        int count = n, i = 0, flag = 0;
        while (count > 0) {
                if (finish[i] == 0) {
                        int status = 1;
                        for (int j=0; j<m; j++)
                                if (need[i][j] > work[j]) {
                                        status = 0;
                                        break;
                                        }
                        if (status == 1) {
                                flag = 1;
                                finish[i] = 1;
                                safety[n-count] = i;
                                count--;
                                for (int j=0; j<m; j++) {
                                        work[j] = work[j] + alloc[i][j];
                                        avail[i][j] = work[j];
                                        }
                                }

                        //printing details of the status of each process
                        printf("\nPID\tAllocated\t\tNeed\t\tStatus\t\tWork\n");
                        printf("%s\t",pid[i]);

                        for (int j=0; j<m; j++)
                                printf("%d ",alloc[i][j]);
                        printf("\t\t\t");

                        for (int j=0; j<m; j++)
                                printf("%d ",need[i][j]);
                        printf("\t-->   ");

                        if (status==1)
                                printf("True\t\t");
```

```
                        else
                                printf("False\t\t");

                        for (int j=0; j<m; j++)
                                printf("%d ",work[j]);

                        printf("\n");
                        }

                i++;

                if (i==n) {
                        if (flag==0 && count!=0) {
                                printf("\nResources could not be allocated for the following processes: \n<");
                                for (int k=0; k<n; k++)
                                        if (finish[k]==0)
                                                printf("%s ",pid[k]);
                                printf("\b>\n");
                                if (choice==4) {
                                        for (int j=0; j<m; j++) {
                                                alloc[reqid][j] -= req[j];
                                                need[reqid][j] += req[j];
                                                }
                                        }
                                //printf("Possibility of deadlock!\n");
                                break;
                                }
                        else {
                                i=0;
                                flag=0;
                                }
                        }
                }

        if (count > 0) {
                if (choice==4)
                        printf("Hence, the request cannot be granted.\n");
                break;
                }

        for (int j=0; j<m; j++)
                work[j] = available[j];

        printf("\nSafety sequence:\n<");
        for (int i=0; i<n; i++)
                printf("%s ",pid[safety[i]]);
        printf("\b>\n");
```

```
                        if (choice==4)
                                printf("Process %s is thus granted.\n",pid[reqid]);

                        break;
                        }

                case 5: {
                        printf("Exiting...\n");
                        exit(0);
                        break;
                        }

                default: {
                        printf("Invalid choice! Enter again...\n");
                        }
                }
        } while (choice!=0);

return 0;
}
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:

```
> ./a7avoid

__BANKER'S ALGORITHM__

Enter no. of processes: 5
Enter no. of resources: 3

Enter details of these resources:-

Resource name: A
    Total no. of instances: 10
Resource name: B
    Total no. of instances: 5
Resource name: C
    Total no. of instances: 7

__MENU__
1. Read data
2. Print data
3. Check system state
4. Resource request
5. Exit


Enter choice: 1

Enter no. of available instances of each resource:-

Resource A: 3
Resource B: 3
Resource C: 2

Enter no. of processes: 5
Enter the details for each process:-

Process name: P0
    Resources allocated: 0 1 0
    Maximum resources needed: 7 5 3
Process name: P1
    Resources allocated: 2 0 0
    Maximum resources needed: 3 2 2
Process name: P2
    Resources allocated: 3 0 2
    Maximum resources needed: 9 0 2
Process name: P3
    Resources allocated: 2 1 1
    Maximum resources needed: 2 2 2
Process name: P4
    Resources allocated: 0 0 2
    Maximum resources needed: 4 3 3
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 3 2
P1  2 0 0       3 2 2       1 2 2
P2  3 0 2       9 0 2       6 0 0
P3  2 1 1       2 2 2       0 1 1
P4  0 0 2       4 3 3       4 3 1
```

```
Enter choice: 3

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     3 3 2

PID Allocated       Need        Status      Work
P1  2 0 0           1 2 2   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P3  2 1 1           0 1 1   -->   True      7 4 3

PID Allocated       Need        Status      Work
P4  0 0 2           4 3 1   -->   True      7 4 5

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   True      7 5 5

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   True      10 5 7

Safety sequence:
<P1 P3 P4 P0 P2>
```

```
Enter choice: 4
Enter pid no. to request: P1
Resources requested: 1 0 2
Process P1:
Available: 2 3 0
Allocation: 3 0 2
Need: 0 2 0

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       2 3 0
P1  3 0 2       3 2 2       0 2 0
P2  3 0 2       9 0 2       6 0 0
P3  2 1 1       2 2 2       0 1 1
P4  0 0 2       4 3 3       4 3 1
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     2 3 0

PID Allocated       Need        Status      Work
P1  3 0 2           0 2 0   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P3  2 1 1           0 1 1   -->   True      7 4 3

PID Allocated       Need        Status      Work
P4  0 0 2           4 3 1   -->   True      7 4 5

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   True      7 5 5

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   True      10 5 7

Safety sequence:
<P1 P3 P4 P0 P2>
Process P1 is thus granted.

Enter choice: 1

Enter no. of available instances of each resource:-

Resource A: 3
Resource B: 3
Resource C: 2

Enter no. of processes: 3
Enter the details for each process:-

Process name: P0
    Resources allocated: 0 1 0
    Maximum resources needed: 7 5 3
Process name: P1
    Resources allocated: 2 0 0
    Maximum resources needed: 3 2 2
Process name: P2
    Resources allocated: 3 0 2
    Maximum resources needed: 9 0 2

Enter choice: 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 3 2
P1  2 0 0       3 2 2       1 2 2
P2  3 0 2       9 0 2       6 0 0
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 3

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     3 3 2

PID Allocated       Need        Status      Work
P1  2 0 0           1 2 2   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

Resources could not be allocated for the following processes:
<P0 P2>

Enter choice: 4
Enter pid no. to request: P1
Resources requested: 1 0 2
Process P1:
Available: 2 3 0
Allocation: 3 0 2
Need: 0 2 0

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       2 3 0
P1  3 0 2       3 2 2       0 2 0
P2  3 0 2       9 0 2       6 0 0

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     2 3 0

PID Allocated       Need        Status      Work
P1  3 0 2           0 2 0   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

Resources could not be allocated for the following processes:
<P0 P2>
Hence, the request cannot be granted.
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 3 2
P1  2 0 0       3 2 2       1 2 2
P2  3 0 2       9 0 2       6 0 0

Enter choice: 4
Enter pid no. to request: P1
Resources requested: 2 2 2
The request cannot be satisfied with the current resources.

Enter choice: 4
Enter pid no. to request: P1
Resources requested: 0 1 0
Process P1:
Available: 3 2 2
Allocation: 2 1 0
Need: 1 1 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 2 2
P1  2 1 0       3 2 2       1 1 2
P2  3 0 2       9 0 2       6 0 0

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     3 2 2

PID Allocated       Need        Status      Work
P1  2 1 0           1 1 2   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

Resources could not be allocated for the following processes:
<P0 P2>
Hence, the request cannot be granted.

Enter choice: 5
Exiting...
```

Name: Krithika Swaminathan
Roll No.: 205001057

2) Algorithm for Deadlock Detection:

Code:

```
//Implementation of Algorithm for Deadlock Detection

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define P_CAP 10
#define R_CAP 5

void safety();

int main() {
	printf("\n__DEADLOCK DETECTION__\n\n");

	int n, m, t=0;
	printf("Enter no. of processes: ");
	scanf("%d",&n);
	printf("Enter no. of resources: ");
	scanf("%d",&m);

	char pid[n][3], rid[m];
	int alloc[n][m], max[n][m], need[n][m], avail[n][m], total[m], available[m], work[m],
finish[n], safety[n];

	printf("\nEnter details of these resources:-\n\n");
	for (int j=0; j<m; j++) {
		printf("Resource name: ");
		scanf(" %c",&rid[j]);
		printf("\tTotal no. of instances: ");
		scanf("%d",&total[j]);
		}

	printf("\n__MENU__\n1. Read data\n2. Print data\n3. Check system state\n4. Exit\n");

	int choice = 1;
	do {
		printf("\nEnter choice: ");
		scanf("%d",&choice);

		switch (choice) {
			case 1: {
				printf("\nEnter no. of available instances of each resource:-\n\n");
				for (int i=0; i<m; i++) {
					printf("Resource %c: ",rid[i]);
					scanf("%d",&avail[0][i]);
					work[i] = avail[0][i];
```

```
                available[i] = avail[0][i];
                }

        printf("\nEnter no. of processes: ");
        scanf("%d",&n);

        printf("Enter the details for each process:-\n\n");
        for (int i=0; i<n; i++) {
                printf("Process name: ");
                scanf("%s",pid[i]);
                printf("\tResources allocated: ");
                for (int j=0; j<m; j++) {
                        scanf("%d",&alloc[i][j]);
                        }
                printf("\tMaximum resources needed: ");
                for (int j=0; j<m; j++) {
                        scanf("%d",&max[i][j]);
                        }
                for (int j=0; j<m; j++) {
                        need[i][j] = max[i][j] - alloc[i][j];
                        }

                finish[i] = 0;
                }

        break;
        }

case 2: {
        //print in table form
        printf("\nPID\tAllocated\tMaximum\t\tNeed\t\tAvailable\n");
        printf("\t");
        for (int col=1; col<=4; col++) {
                for (int j=0; j<m; j++)
                        printf("%c ",rid[j]);
                printf("\t\t");
                }
        printf("\n");

        for (int i=0; i<n; i++) {
                printf("%s\t",pid[i]);

                for (int j=0; j<m; j++)
                        printf("%d ",alloc[i][j]);
                printf("\t\t");

                for (int j=0; j<m; j++)
                        printf("%d ",max[i][j]);
                printf("\t\t");
```

```
                for (int j=0; j<m; j++)
                        printf("%d ",need[i][j]);
                printf("\t\t");

                if (i==0) {
                        for (int j=0; j<m; j++)
                                printf("%d ",available[j]);
                        }

                printf("\n");
                }
        break;
        }

case 3: {
        //display current status
        if (choice==3) {
                for (int i=0; i<n; i++) {
                        for (int j=0; j<m; j++) {
                                need[i][j] = max[i][j] - alloc[i][j];
                                }
                        }
                }

        for (int k=0; k<n; k++)
                finish[k] = 0;

        printf("\nChecking the status of the set of processes:- \n");

        int count = n, i = 0, flag = 0;
        while (count > 0) {
                if (finish[i] == 0) {
                        int status = 1;
                        for (int j=0; j<m; j++)
                                if (need[i][j] > work[j]) {
                                        status = 0;
                                        break;
                                        }
                        if (status == 1) {
                                flag = 1;
                                finish[i] = 1;
                                safety[n-count] = i;
                                count--;
                                for (int j=0; j<m; j++) {
                                        work[j] = work[j] + alloc[i][j];
                                        avail[i][j] = work[j];
                                        }
                                }
```

```
                        //printing details of the status of each process
                        printf("\nPID\tAllocated\t\tNeed\t\tStatus\t\tWork\n");
                        printf("%s\t",pid[i]);

                        for (int j=0; j<m; j++)
                                printf("%d ",alloc[i][j]);
                        printf("\t\t\t");

                        for (int j=0; j<m; j++)
                                printf("%d ",need[i][j]);
                        printf("\t-->   ");

                        if (status==1)
                                printf("True\t\t");
                        else
                                printf("False\t\t");

                        for (int j=0; j<m; j++)
                                printf("%d ",work[j]);

                        printf("\n");
                        }

                i++;

                if (i==n) {
                        if (flag==0 && count!=0) {
                                printf("\nPossibility of deadlock!\n");
                                printf("\nThe following processes may cause the
deadlock: \n<");

                                for (int k=0; k<n; k++)
                                        if (finish[k]==0)
                                                printf("%s ",pid[k]);
                                printf("\b>\n");
                                break;
                                }
                        else {
                                i=0;
                                flag=0;
                                }
                        }
                }

        if (count > 0)
                break;

        for (int j=0; j<m; j++)
                work[j] = available[j];
```

```
                        printf("\nSafety sequence:\n<");
                        for (int i=0; i<n; i++)
                                printf("%s ",pid[safety[i]]);
                        printf("\b>\n");

                        break;
                        }

                case 4: {
                        printf("Exiting...\n");
                        exit(0);
                        break;
                        }

                default: {
                        printf("Invalid choice! Enter again...\n");
                        }
                }
        } while (choice!=0);

return 0;
}
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:

```
> ./a7detect

__DEADLOCK DETECTION__

Enter no. of processes: 5
Enter no. of resources: 3

Enter details of these resources:-

Resource name: A
    Total no. of instances: 10
Resource name: B
    Total no. of instances: 5
Resource name: C
    Total no. of instances: 7

__MENU__
1. Read data
2. Print data
3. Check system state
4. Exit

Enter choice: 1

Enter no. of available instances of each resource:-

Resource A: 3
Resource B: 3
Resource C: 2

Enter no. of processes: 5
Enter the details for each process:-

Process name: P0
    Resources allocated: 0 1 0
    Maximum resources needed: 7 5 3
Process name: P1
    Resources allocated: 2 0 0
    Maximum resources needed: 3 2 2
Process name: P2
    Resources allocated: 3 0 2
    Maximum resources needed: 9 0 2
Process name: P3
    Resources allocated: 2 1 1
    Maximum resources needed: 2 2 2
Process name: P4
    Resources allocated: 0 0 2
    Maximum resources needed: 4 3 3
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 3 2
P1  2 0 0       3 2 2       1 2 2
P2  3 0 2       9 0 2       6 0 0
P3  2 1 1       2 2 2       0 1 1
P4  0 0 2       4 3 3       4 3 1
```

```
Enter choice: 3

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     3 3 2

PID Allocated       Need        Status      Work
P1  2 0 0           1 2 2   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P3  2 1 1           0 1 1   -->   True      7 4 3

PID Allocated       Need        Status      Work
P4  0 0 2           4 3 1   -->   True      7 4 5

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   True      7 5 5

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   True      10 5 7

Safety sequence:
<P1 P3 P4 P0 P2>
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
Enter choice: 1

Enter no. of available instances of each resource:-

Resource A: 3
Resource B: 3
Resource C: 2

Enter no. of processes: 3
Enter the details for each process:-

Process name: P0
    Resources allocated: 0 1 0
    Maximum resources needed: 7 5 3
Process name: P1
    Resources allocated: 2 0 0
    Maximum resources needed: 3 2 2
Process name: P2
    Resources allocated: 3 0 2
    Maximum resources needed: 9 0 2

Enter choice: 2

PID Allocated   Maximum     Need        Available
    A B C       A B C       A B C       A B C
P0  0 1 0       7 5 3       7 4 3       3 3 2
P1  2 0 0       3 2 2       1 2 2
P2  3 0 2       9 0 2       6 0 0

Enter choice: 3

Checking the status of the set of processes:-

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     3 3 2

PID Allocated       Need        Status      Work
P1  2 0 0           1 2 2   -->   True      5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

PID Allocated       Need        Status      Work
P0  0 1 0           7 4 3   -->   False     5 3 2

PID Allocated       Need        Status      Work
P2  3 0 2           6 0 0   -->   False     5 3 2

Possibility of deadlock!

The following processes may cause the deadlock:
<P0 P2>

Enter choice: 4
Exiting...
```

Name: Krithika Swaminathan
Roll No.: 205001057

Learning outcomes:

- The Banker's Algorithm for deadlock avoidance was understood and implemented.
- A safety sequence for process synchronization was obtained for a set of given processes and available resources.
- Methods for avoiding and detecting deadlocks were implemented.
