Deadlock Detection and Avoidance

-B.Ajay
21011102020

Aim:

To write a program to

1.Avoid Deadlocks (Banker 's Algorithm)
2.Detect Deadlocks

Algorithm:

1.**Deadlock Avoidance (Banker's Algorithm):**

1. Accept all the required matrices that is allocation,max,available,given
2. Compute the need matrix using max[i][j]-alloc[i][j]
3. Keep a visited array to keep track of sequence
4. Traverse through need array if need[i][j]<=available[i][j] then we raise flag and we update our vis array then we mark the need array cost as inf to again traverse the matrix till we get all n process
5. If the above is not possible and result in deadlock print no safe state possible
6. Finally print safe state from visited array

2. **Deadlock Detection:**

1. Accept all the required matrices that is allocation,request,available
2. Traverse through required array if request[i][j]<=available[i][j] then we raise flag and we update our vis array then we mark the need array cost as inf to again traverse the matrix till we get all n process and add available[i][*] with alloc[i][*] as the new available
3. if no flag raised during iteration that is no condition met then system is in deadlock
4. If not we print no deadlock detected after the whole iteration finished

Code:

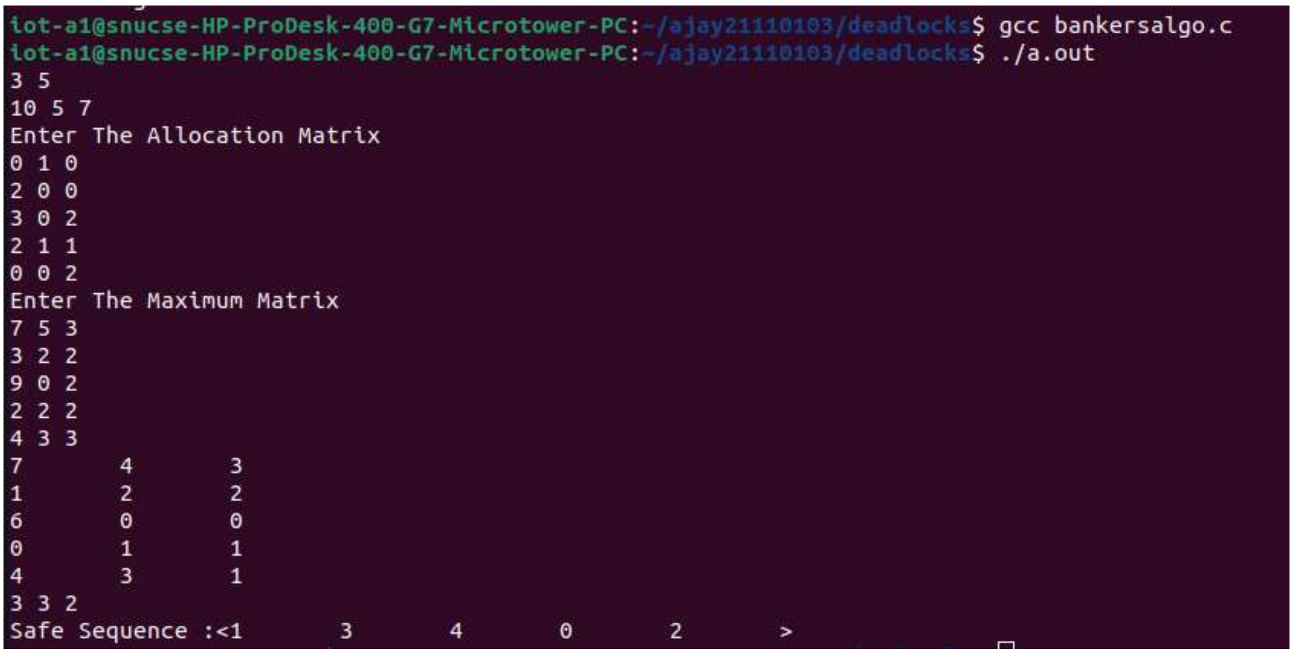
BankersAlgo.c:

```
#include <stdio.h>
int main(){
int n_res,tmp,n_proc;
scanf("%d %d",&n_res,&n_proc);
int given[1][n_res],m=0;
int
allocation_matrix[n_proc][n_res],max[n_proc][n_res],available[n_proc][n_res],need[n_proc][n_res]
;
int vis[n_proc];
for(int i=0;i<n_res;i++){
        scanf("%d",&tmp);
        given[0][i]=tmp;
}
printf("Enter The Allocation Matrix\n");
for(int i=0;i<n_proc;i++){
        for(int j=0;j<n_res;j++){
                scanf("%d",&tmp);
                allocation_matrix[i][j]=tmp;
        }
}

printf("Enter The Maximum Matrix\n");
```

```
for(int i=0;i<n_proc;i++){
	for(int j=0;j<n_res;j++){
		scanf("%d",&tmp);
		max[i][j]=tmp;
	}
}

for(int i=0;i<n_proc;i++){
	for(int j=0;j<n_res;j++){
		need[i][j]=max[i][j]-allocation_matrix[i][j];
	}

}

for(int i=0;i<n_proc;i++){
	for(int j=0;j<n_res;j++){
		printf("%d\t",need[i][j]);
	}
	printf("\n");
}


for(int i=0;i<n_res;i++){
	scanf("%d",&tmp);
	available[0][i]=tmp;
}
int f1=0,f2=0,f3=0;
int avl_cp[1][n_res];
//for(int i=0;i<n_proc;i++){
while(m<n_proc){
	for(int j=0;j<n_proc;j++){
		for(int k=0;k<n_res;k++){
			if(available[0][k]>=need[j][k] ){f1++;}
			else{break;}
		}
		if(f1==3){
			f1=0;
			f2=1;
			for(int p=0;p<n_res;p++){
				need[j][p]=10203123;
			}
			for(int o=0;o<n_res;o++){
				avl_cp[0][o]=available[0][o];
			}
//			printf("%d\n",j);
			vis[m++]=j;
			for(int l=0;l<n_res;l++){
```

```
                        available[0][l]=avl_cp[0][l]+allocation_matrix[j][l];
                    }
                }
            }
        if(f2!=1){
                printf("No Safe Sequence Possible\n");
                return -1;
        }
}
printf("Safe Sequence :<");
for(int i=0;i<n_proc;i++){printf("%d\t",vis[i]);}
printf(">\n");
return 1;
}
```

Output:



```
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/deadlocks$ gcc bankersalgo.c
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/deadlocks$ ./a.out
3 5
10 5 7
Enter The Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter The Maximum Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
7       4       3
1       2       2
6       0       0
0       1       1
4       3       1
3 3 2
Safe Sequence :<1       3       4       0       2       >
```

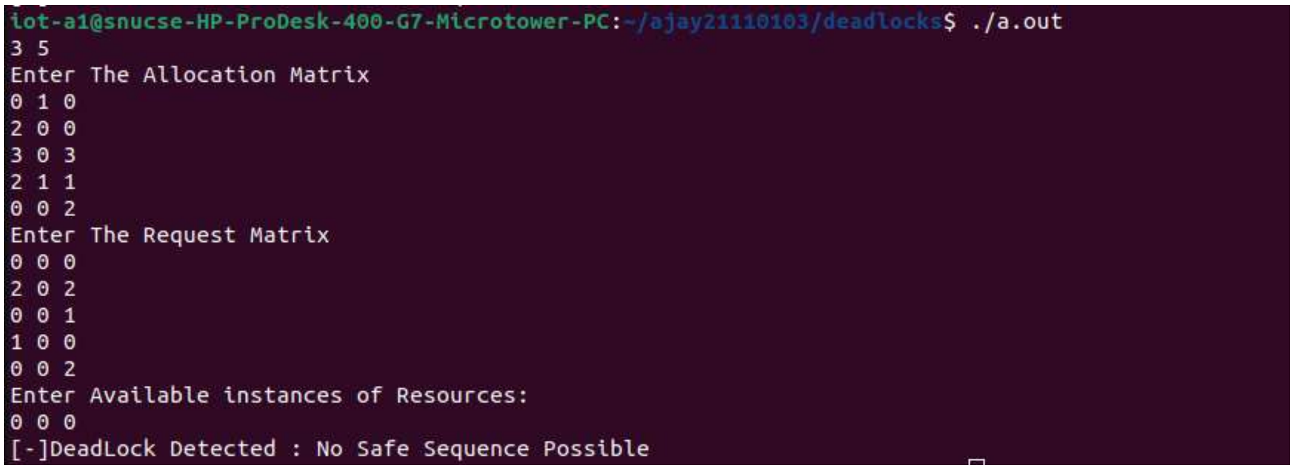
2. **DeadlockDetection.c**

```
#include <stdio.h>
int main(){
int n_res,tmp,n_proc;
scanf("%d %d",&n_res,&n_proc);
int m=0;
int allocation_matrix[n_proc][n_res],request[n_proc][n_res],available[n_proc][n_res];
printf("Enter The Allocation Matrix\n");
for(int i=0;i<n_proc;i++){
        for(int j=0;j<n_res;j++){
                scanf("%d",&tmp);
```

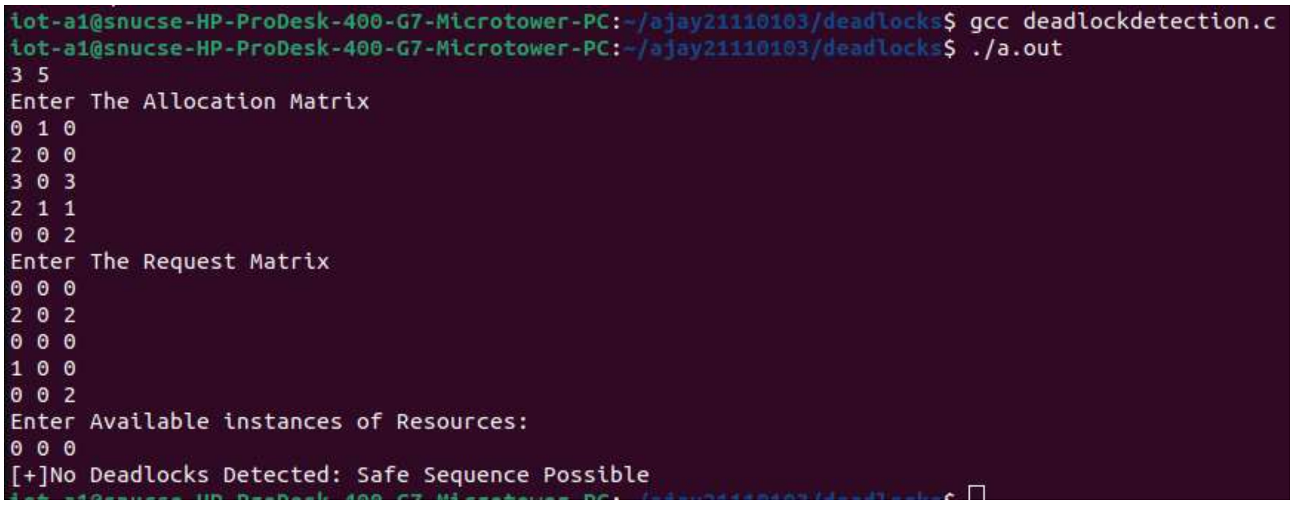
```
				allocation_matrix[i][j]=tmp;
			}
		}
		printf("Enter The Request Matrix\n");


		for(int i=0;i<n_proc;i++){
			for(int j=0;j<n_res;j++){
				scanf("%d",&tmp);
				request[i][j]=tmp;
			}
		}
		printf("Enter Available instances of Resources: \n");
		for(int i=0;i<n_res;i++){
			scanf("%d",&tmp);
			available[0][i]=tmp;
		}
		int f1=0,f2=0,f3=0;
		int avl_cp[1][n_res];
		while(m<n_proc){
			for(int j=0;j<n_proc;j++){
				for(int k=0;k<n_res;k++){
					if(available[0][k]>=request[j][k] ){f1++;}
					else{break;}
				}
				if(f1==3){
					f1=0;
					f2=1;
					for(int p=0;p<n_res;p++){
						request[j][p]=10203123;
					}
					for(int o=0;o<n_res;o++){
						avl_cp[0][o]=available[0][o];
					}
//					printf("%d\n",j);
					m++;
					for(int l=0;l<n_res;l++){
						available[0][l]=avl_cp[0][l]+allocation_matrix[j][l];
					}
				}
			}
			if(f2!=1){
				printf("[-]DeadLock Detected : No Safe Sequence Possible\n");
				return -1;
			}
			f2=0;
		}
		printf("[+]No Deadlocks Detected: Safe Sequence Possible\n");
		return 1;
```

}

Output:



```
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/deadlocks$ ./a.out
3 5
Enter The Allocation Matrix
0 1 0
2 0 0
3 0 3
2 1 1
0 0 2
Enter The Request Matrix
0 0 0
2 0 2
0 0 1
1 0 0
0 0 2
Enter Available instances of Resources:
0 0 0
[-]DeadLock Detected : No Safe Sequence Possible
```



```
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/deadlocks$ gcc deadlockdetection.c
iot-a1@snucse-HP-ProDesk-400-G7-Microtower-PC:~/ajay21110103/deadlocks$ ./a.out
3 5
Enter The Allocation Matrix
0 1 0
2 0 0
3 0 3
2 1 1
0 0 2
Enter The Request Matrix
0 0 0
2 0 2
0 0 0
1 0 0
0 0 2
Enter Available instances of Resources:
0 0 0
[+]No Deadlocks Detected: Safe Sequence Possible
```

Result:

Thus the deadlock avoidance and detection algorithms were implemented and studied