Name – Ashutosh Ardu

Regno – 20BRS1262

OS Lab 7

Banker's Algorithm

Given Process Sequence

	Allocation				*Max*				*Need*				*Available*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	2	0	1	1	3	2	1	1					6	4	4	2
P_1	1	1	0	0	1	2	0	2								
P_2	1	0	1	0	3	2	1	0								
P_3	0	1	0	1	2	1	0	1								

1. Consider the following snapshot of a system in which four resources A, B, C and D are available. The system contains a total of 6 instances of A, 4 of resource B, 4 of resource C, 2 resources D.

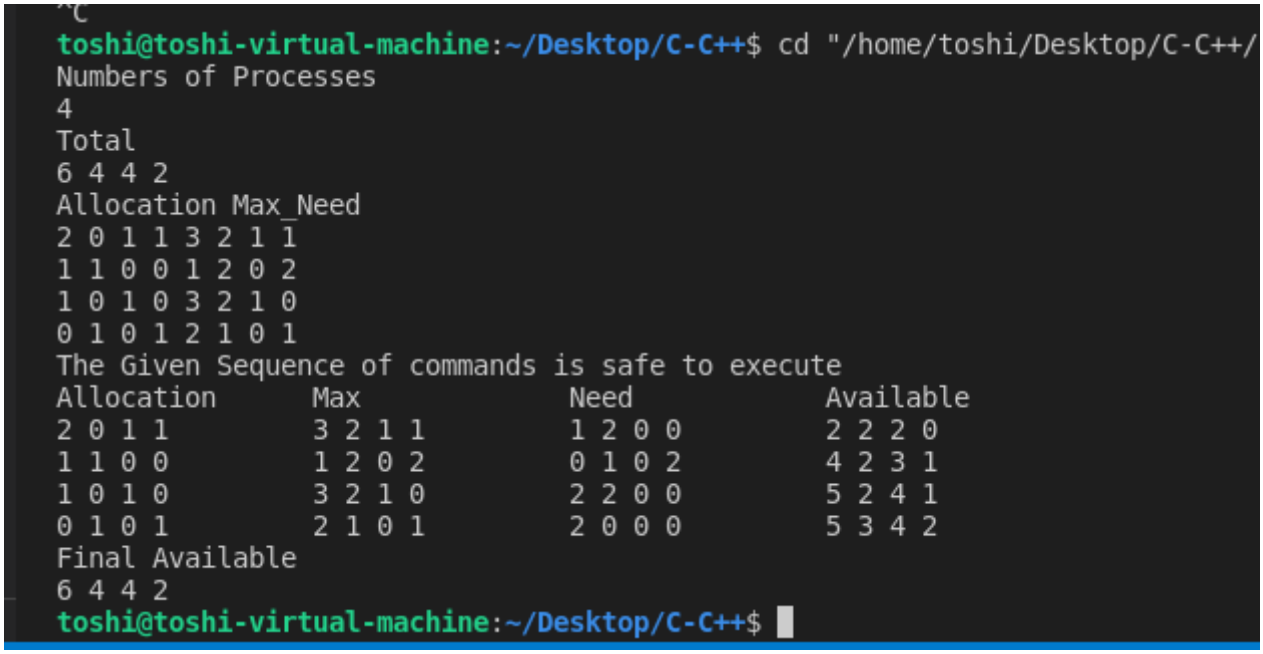
Code

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

struct job{
    int a,b,c,d,finish;
}avail;
int comp(struct job a,struct job *b){
    if(a.a<=b->a && a.b<=b->b && a.c<=b->c && a.d<=b->d) return 1;
    return 0;
}

void sum(struct job *r,struct job *proc){
    r->a+=proc->a;
    r->b+=proc->b;
    r->c+=proc->c;
    r->d+=proc->d;
}
```

```
void get(struct job *proc,int a){
    scanf("%d %d %d %d",&proc->a,&proc->b,&proc->c,&proc->d);
    proc->finish=0;
    if(a==1){
        sum(&avail,proc);
    }
}

void assign(struct job a,struct job proc[],int i){
    proc[i].a=a.a;
    proc[i].b=a.b;
    proc[i].c=a.c;
    proc[i].d=a.d;
}

void put(struct job *x,struct job *y,struct job *z){
    x->a=z->a-y->a;
    x->b=z->b-y->b;
    x->c=z->c-y->c;
    x->d=z->d-y->d;
    x->finish=0;
}

void append(struct job *a,struct job b){
    a->a+=b.a;
    a->b+=b.b;
    a->c+=b.c;
    a->d+=b.d;
}

int check(struct job remain[],struct job *a,struct job b[],struct job current
[],int *t,int n){
    for(int i=0;i<n;++i){
        if(!remain[i].finish){
            if(comp(remain[i],a)){
                remain[i].finish=1;
                append(a,b[i]);
                assign(avail,current,*t);*t+=1;
                return 1;
            }
        }
    }return 0;
}

void print(struct job a[],struct job b[],struct job c[],struct job d[],int n)
{
    printf("Allocation\tMax\t\tNeed\t\tAvailable\n");
    for(int i=0;i<n;++i){
    printf("%d %d %d %d\t\t",a[i].a,a[i].b,a[i].c,a[i].d);
    printf("%d %d %d %d\t\t",b[i].a,b[i].b,b[i].c,b[i].d);
    printf("%d %d %d %d\t\t",c[i].a,c[i].b,c[i].c,c[i].d);
    printf("%d %d %d %d\n",d[i].a,d[i].b,d[i].c,d[i].d);
    }
```

```
}

void copy(struct job a[],struct job b[],int i){
    a[i].a=b[i].a;
    a[i].b=b[i].b;
    a[i].c=b[i].c;
    a[i].d=b[i].d;
}

int main(){
    printf("Numbers of Processes\n");
    int n,s=0,t=0;scanf("%d",&n);
    struct job need[n],total,remain[n],alloc[n],current[n],alloc1[n];
    avail.a=avail.b=avail.c=avail.d=0;
    printf("Total\n");get(&total,0);
    printf("Allocation Max_Need\n");
    for(int i=0;i<n;++i){
        get(&alloc[i],1);
        copy(alloc1,alloc,i);
        get(&need[i],0);
        put(&remain[i],&alloc[i],&need[i]);
    }avail.a=total.a-avail.a;
    avail.b=total.b-avail.b;
    avail.c=total.c-avail.c;
    avail.d=total.d-avail.d;
    printf("%d %d %d %d\n",avail.a,avail.b,avail.c,avail.d);
    assign(avail,current,t);t++;
    for(int i=0;i<n;++i){
        if(!(check(remain,&avail,alloc,current,&t,n))){
            printf("Deadlock Occured\n");
            break;s=1;
        }
    }
    if(s==0){
        printf("The Given Sequence of commands is safe to execute\n");
        print(alloc1,need,remain,current,n);
        printf("Final Available\n");
        printf("%d %d %d %d\n",avail.a,avail.b,avail.c,avail.d);
    }
}
```

Output



Conclusion

The Sequence is safe to execute and will not cause any deadlock.