EX:NO:7 Bankers Algorithm for Deadlock avoidance

Aim :

To implement and study Bankers Algorithm for Deadlock Avoidance Problem

Program :

Banker's Algorithm

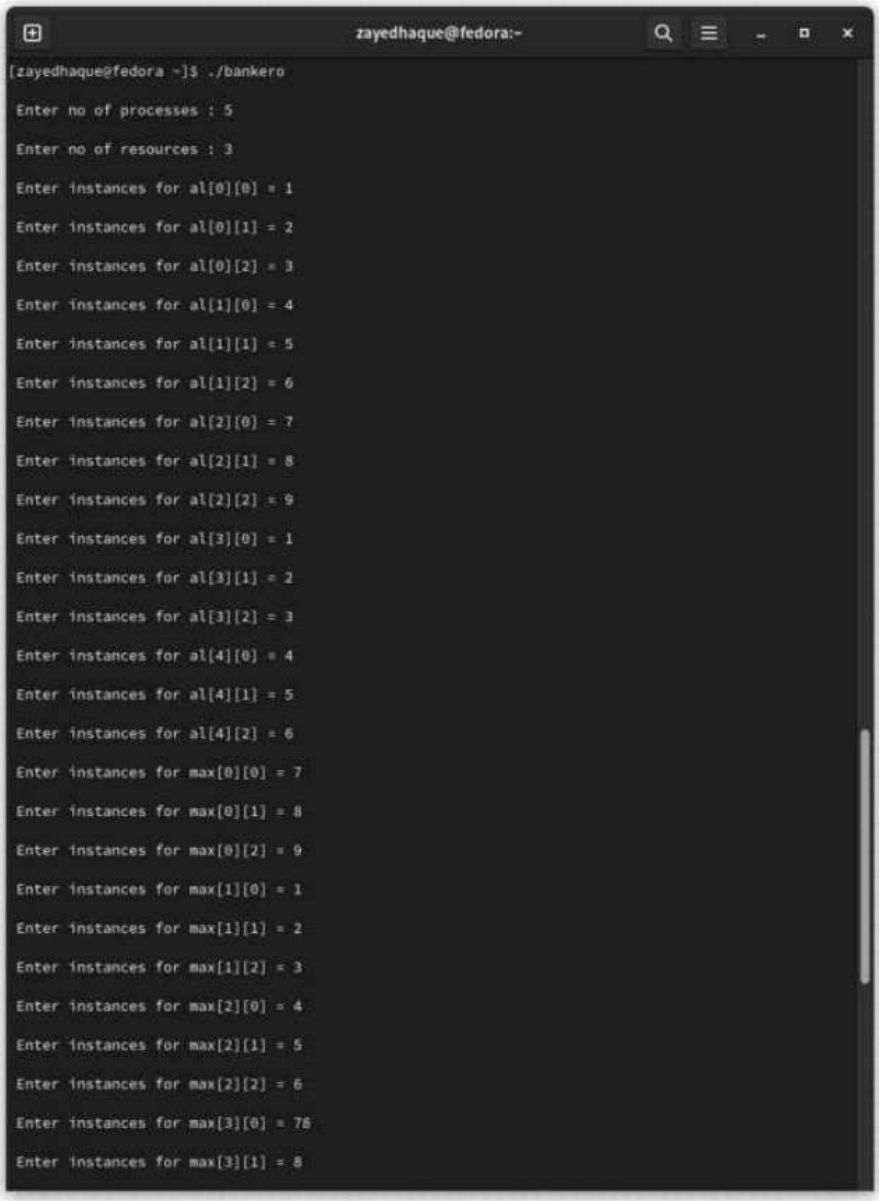
```
#include <stdio.h>

int m, n, i, j, al[10][10], max[10][10], av[10], need[10][10], temp, z, y, p, k;
void main() {

printf("\n Enter no of processes : ");
scanf("%d", &m); // enter numbers of processes

printf("\n Enter no of resources : ");
scanf("%d", &n); // enter numbers of resources

for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
printf("\n Enter instances for al[%d][%d] = ", i,j); // al[][] matrix is for allocated
instances scanf("%d", &al[i][j]);
 al[i][j]=temp;
}
}

for (i = 0; i < m;
i++) {
for (j = 0; j < n; j++) {printf("\n Enter instances for max[%d][%d] = ", i,j); // max[][]
matrix is for max instances
scanf("%d", &max[i][j]);
}
}

for (i = 0; i < n; i++) {
printf("\n Available Resource for av[%d] = ",i); // av[] matrix is for available instances
scanf("%d", &av[i]);
        }
```

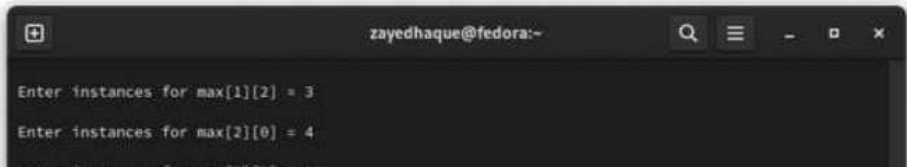
```
        // Print allocation values
printf("Alocation Values :\n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {printf(" \t %d", al[i][j]); // printing allocation matrix
        }
printf("\n");
}

printf("\n\n");

// Print max values
printf("Max Values :\n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
printf(" \t %d", max[i][j]); // printing max matrix
}
printf("\n");
}

printf("\n\n");

// Print need values
printf("Need Values :\n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
need[i][j] = max[i][j] - al[i][j]; // calculating need matrix
printf("\t %d", need[i][j]);      //  printing need matrix
}
printf("\n");
}
p = 1; // used for terminating while loop
y = 0;
while (p != 0) {
for (i = 0; i < m; i++) {
z = 0;
for (j = 0; j < n; j++) {
if (need[i][j] <= av[j]  &&
(need[i][0] != -1)) { // comparing need with available instance and
// checking if the process is done
// or not
z++;                // counter if condition TRUE
}
        }
if (z == n) { // if need<=available TRUE for all resources then condition
// is TRUE
for (k = 0; k < n; k++) {
av[k] += al[i][k]; // new work = work + allocated
}
```

```
}
```

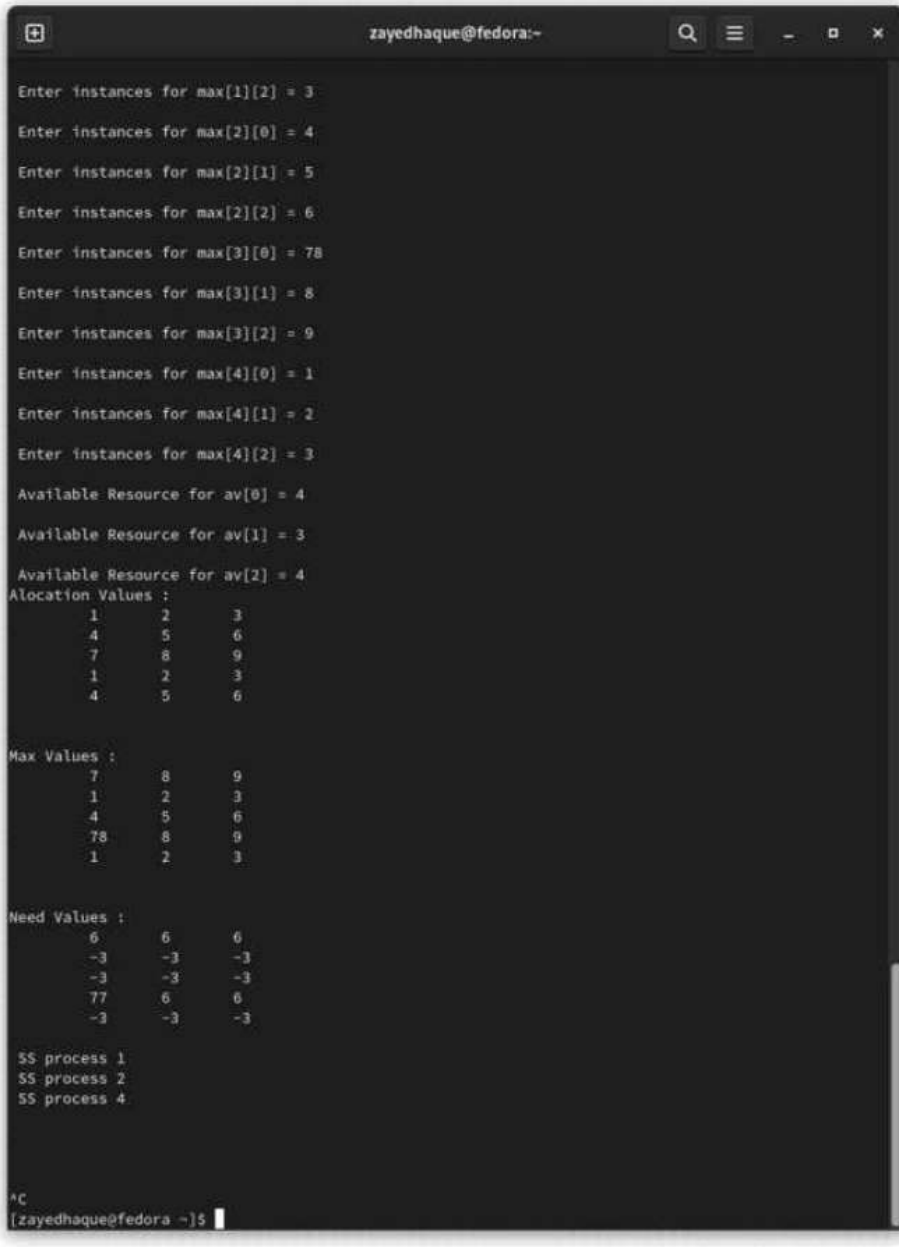
```
printf("\n SS process %d", i); // Print the Process
need[i][0] = -1;              // assign -1 if Process done
y++;                          // cont if process done
}

} // end for loop

if (y == m) { // if all done then
p = 0;      // exit while loop
}} // end while printf("\n");
          }
```

Output :





```
Enter instances for max[1][1] = 2

Enter instances for max[1][2] = 3

Enter instances for max[2][0] = 4

Enter instances for max[2][1] = 5

Enter instances for max[2][2] = 6

Enter instances for max[3][0] = 78

Enter instances for max[3][1] = 8
```



Result :

Successfully implemented the concepts of Banker's Algorithm.

EX:NO:10 FIFO Page Replacement Algorithm

Aim :

To implement FIFO page replacement algorithm

Program :

FIFO Page Replacement Algorithm

```
#include <stdio.h>
int a[100],b[100],i,n,z,f,j,pf,h,temp;
void main(){
printf("\nEnter the no. of pages : ");          // no. of page
referencing scanf("%d",&n);
printf("\nEnter the size of frame : ");         // no. of page
frames scanf("%d",&f);
printf("\n Enter the pages value :\n");              // values of page
referencing for(i=0;i<n;i++){
scanf("%d",&a[i]);
}
for(i=0;i<f;i++){                                                  // assign values of
page frames 1 innitially
b[i]=-1;
}
i=0;j=0;h=0; // i , j used for loop, h for hit count all initialized to 0
while(i<n){
if(b[i]=-1 && i<f ){   // when frames are empty so for starting
enquee b[i]=a[i];
pf++;                                  // page fault counter
}
else
{
z=0
;
for(j=0;j<f;j++){

        if(b[j]==a[i]){                // to check if value already
        present h++;                        // hit counter
        }
        else{
        z++;                           // if not hit count increment
        }
}
if(z==f){                                      // if no value
        matched pf++;                               // page
        fault counter for(j=0;j<f-1;j++){          // shifting
        values
```

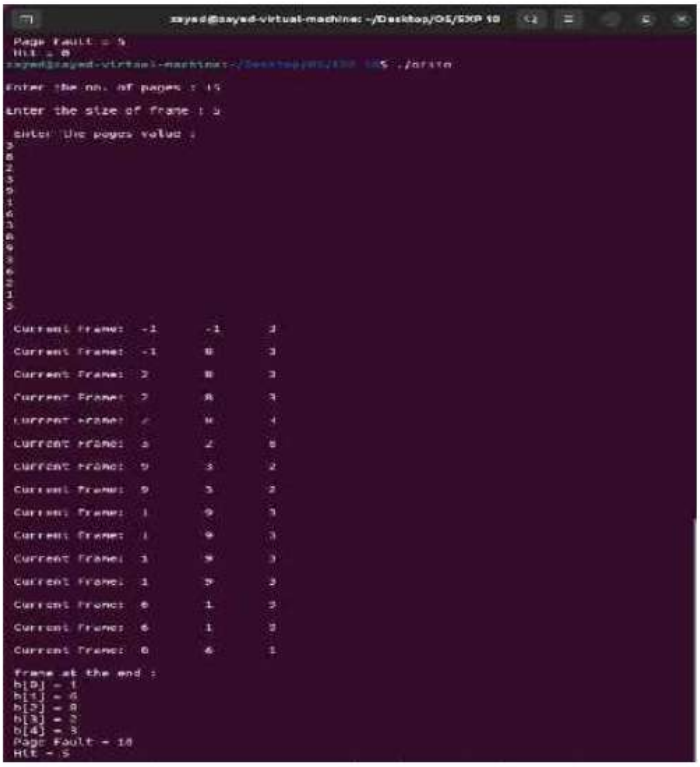
```
        temp=b[j];
        b[j]=b[j+1]
        ;
        b[j+1]=tem
        p;
        }
```

```
        values
```

```
        temp=b[j];
        b[j]=b[j+1]
        ;
        b[j+1]=tem
        p;
        }
        b[f-1]=a[i];                    // insert new values
}
} // end else
printf("\n Current Frame:  %d \t %d \t %d \n",b[2],b[1],b[0]);      // frames value for
every iteration
i++;
}       //end while
printf("\n frame at the end :");
for(i=0;i<f;i++){
printf("\n b[%d] = %d",i,b[i]);              // frame values at the end
}
printf("\n Page Fault = %d ",pf);            // no. of page
faults printf("\n Hit = %d ",h);                                    //
no. of hitS
printf("\n");




}
```

Output :

Output :



Result :

Successfully implemented page replacement using FIFO algorithm

EX:NO:11 LRU AND LFU Page Replacement Algorithm