NATIONAL INSTITUTE OF TECHNOLOGY

KARNATAKA

OPERATING SYSTEM LAB

EXPERIMENT :7

Name : Shivam Yadav

Roll No : 211CS257

Section : S2

Code :CS257

Q) Write code for deadlock detection for following two cases:

i)single instance resource type

ii)multiple instance resource type

part 1: code for single instance resource type

```
#include<stdio.h>

//global declaration
int n,wfg[10][10];

//function for cycle check
int check_cycle(int process, int visit[], int stack[])
{
   visit[process]=1;
   stack[process]=1;
   for(int i=0;i<n;i++)
   {
            if(wfg[process][i])
            {
               if(visit[i]==0&&check_cycle(i,visit,stack))
                    return 1;
               if(stack[i]==1)
                   return 1;
            }
   }
   stack[process]=0;
   return 0;
}
```

```
//deadlock function
int deadlock()
{
   int visit[n];
   for(int i=0;i<n;i++)
          visit[i]=0;

   for(int i=0;i<n;i++)
   {
      if(visit[i]==0)
      {
           int stack[n];
           for(int t=0;t<n;t++)
                    stack[t]=0;
           if(check_cycle(i,visit,stack))
                     return 1;
      }
   }
   return 0;
}

int main()
{
   printf("number of processes: ");
   scanf("%d",&n);

   printf("number of resources: ");
   int m;
   scanf("%d",&m);
```

```
int rag[n][m];
for(int i=0;i<n;i++)
{
        for(int j=0;j<m;j++)
                rag[i][j]=0;
}

for(int i=0;i<n;i++)
{
            for(int j=0;j<n;j++)
                wfg[i][j]=0;
}

printf("enter edges in the graph:\n");
printf("R  P: enter 0:\n");
printf("R->P: enter 1:\n");
printf("P->R: enter -1:\n");

for(int i=0;i<n;i++)
{
   for(int j=0;j<m;j++)
      scanf("%d",&wfg[i][j]);
}
```

```
    for(int i=0;i<n;i++)
    {
       for(int j=0;j<m;j++)
       {
          for(int k=0;k<n;k++)
          {
             if(rag[i][j]==-1&&rag[j][k]==1)
                      wfg[i][k]=1;
          }
       }
    }

    if(deadlock()==1)
            printf("deadlock detected\n");
    else
           printf("no deadlock detected\n");
    return 0;
}
```

Output :

For first input:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop/211cs257$ ./a.out
number of processes: 4
number of resources: 4
enter edges in the graph:
R  P: enter 0:
R->P: enter 1:
P->R: enter -1:
1 -1 0 0
0 1 -1 0
0 0 1 -1
-1 0 0 1
deadlock detected
```

For second input:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop/211cs257$ ./a.out
number of processes: 4
number of resources: 4
enter edges in the graph:
R  P: enter 0:
R->P: enter 1:
P->R: enter -1:
1 -1 0 0
0 1 -1 0
0 0 -1 -1
-1 0 0 1
no deadlock detected
```

Part 2: code for multiple resource type

```
#include<bits/stdc++.h>

using namespace std;


int main()

{

	vector<int> ans;


	//number of process

	int n=3;


	//types of resources

	int m=3;


	//allocated resources in allocated matrix

	int tot_allocated[m];

	for(int i=0;i<m;++i)

		tot_allocated[i]=0;


	int allocated[n][m]={{1,2,1},{2,0,1},{2,2,1}};


	//maximum requirement by each process in max matrix

	int max[n][m]={{2,2,4},{2,1,3},{3,4,1}};


	//total resources in system

	int total[m]={5,5,5};
```

```
//calculation of need matrix
int need[n][m];


for(int i=0;i<n;++i)
{
	for(int j=0;j<m;++j)
	{
		need[i][j] = max[i][j] - allocated[i][j] ;



		tot_allocated[j]+=allocated[i][j];
	}
}

//eval available resources av=tot-allo;
int available[m];
for(int i=0;i<m;++i)
	available[i] = total[i]-tot_allocated[i] ;

queue<int> q;
for(int i=0;i<n;++i)
	q.push(i);
```

```
while(q.size()!=0)
{
	int i=q.front();
	int flag =1;
	for(int j=0;j<m;++j)
	{
		if(need[i][j]>available[j])
		{
			flag=0;
			break;
		}
	}

	if(flag==1)
	{
		//update available resources
		for(int j=0;j<m;++j)
		{
			available[j]+=allocated[i][j];
		}

		//push in ans
		ans.push_back(i);

		//remove that process from queue
		q.pop();
	}
```

```
            else
            {
                  //delete from front add to back of queue
                  q.pop();

                  q.push(i);
            }
      }

      cout<<"\nsystem in safe state\n";
      cout<<"safe sequence is :";

      int k=ans.size();
      for(int i=0;i<k-1;++i)
            cout<<"p"<<ans[i]<<"->";

      cout<<"p"<<ans[k-1]<<endl;

}
```

Output :

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop/211cs257$ ./a.out

system in safe state
safe sequence is :p1->p0->p2
```