Week - 6
(27 July 2023)
Experiment - 6

Question:

1) Write a C program to simulate Banker's Algorithm for the purpose of deadlock avoidance.

2) Write a C program to simulate deadlock detection.

Program:

1) Banker's Algorithm:

```
#include <stdio.h>

int main()
{
   int n, m, i, j, k;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   printf("Enter the number of resources: ");
   scanf("%d", &m);

   int allocation[n][m];
   printf("Enter the Allocation Matrix:\n");
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
      {
         scanf("%d", &allocation[i][j]);
      }
   }

   int max[n][m];
   printf("Enter the MAX Matrix:\n");
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < m; j++)
      {
         scanf("%d", &max[i][j]);
      }
   }

   int available[m];
   printf("Enter the Available Resources:\n");
   for (i = 0; i < m; i++)
   {
      scanf("%d", &available[i]);
   }

   int f[n], ans[n], ind = 0;
   for (k = 0; k < n; k++)
   {
      f[k] = 0;
   }
```

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

int y = 0;
for (k = 0; k < n; k++)
{
   for (i = 0; i < n; i++)
   {
      if (f[i] == 0)
      {
         int flag = 0;
         for (j = 0; j < m; j++)
         {
            if (need[i][j] > available[j])
            {
               flag = 1;
               break;
            }
         }

         if (flag == 0)
         {
            ans[ind++] = i;
            for (y = 0; y < m; y++)
            {
               available[y] += allocation[i][y];
            }
            f[i] = 1;
         }
      }
   }
}

int flag = 1;
for (i = 0; i < n; i++)
{
   if (f[i] == 0)
   {
      flag = 0;
      printf("The following system is not safe\n");
      break;
   }
}

if (flag == 1)
{
   printf("Following is the SAFE Sequence\n");
   for (i = 0; i < n - 1; i++)
   {
```

```
            printf(" P%d ->", ans[i]);
        }
        printf(" P%d\n", ans[n - 1]);
    }
    return 0;
}
```

Output: (Banker's Algorithm)

```
Enter the number of processes: 5
Enter the number of resources: 3
Enter the Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the MAX Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Available Resources:
3 3 2
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
```

```
Enter the number of processes: 5
Enter the number of resources: 3
Enter the Allocation Matrix:
0 2 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the MAX Matrix:
8 4 6
3 5 7
3 6 7
9 5 3
2 5 7
Enter the Available Resources:
3 2 2
The following system is not safe
```

Observation Book Pictures: (Banker's Algorithm)

PAGE NO :
DATE : 27/07/2023

Experiment - 8

Write a c program to simulate Banker's algorithm for the purpose of deadlock avoidance:

Program:

```
#include <stdio.h>
int main()
{ int n, m, i, j, k;
  printf(" Enter the number of processes: ");
  scanf("%d", &n);
  printf(" Enter the number of resources: ");
  scanf("%d", &m);

  int allocation[n][m];
  printf(" Enter the allocation matrix :\n");
  for(i=0; i<n; i++)
  { for(j=0; j<m; j++)
    { scanf("%d", &allocation[i][j]);
    }
  }

  int max[n][m];
  printf(" Enter the max matrix: \n");
  for(i=0; i<n; i++)
  { for(j=0; j<m; j++)
    { scanf("%d", &max[i][j]);
    }
  }
```

→

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

int f[n], ans[n], ind=0;
for(k=0; k<n; k++)
  { f[k]=0;
  }

int y=0;
for(k=0; k<n; k++)
  { for(i=0; i<n; i++)
    { if(f[i]==0)
      { int flag=0;
        for(j=0; j<m; j++)
        { if(need[i][j] > available[j])
          { flag=1;
            break;
          }
        }
        if(flag==0)
        { ans[ind++]=i;
          for(y=0; y<m; y++)
          { available[y] += allocation[i][y];
          }
          f[i]=1;
        }
      }
  }
}
```

```
int flag = 1;
for(i=0; i<n; i++)
  {if (f[i] == 0)
    { flag = 0;
      printf("The following system is not safe \n");
      break;
    }
  }

if (flag == 1)
{ printf("Following is the Safe Sequence : \n");
  for( i=0; i<n-1; i++)
    { printf("P %d ->", ans[i]);
    }
  printf("P%d \n", ans[n-1]);
}
  return 0;
}
```

Output:

Enter the number of processes : 5

Enter the number of resources : 3

Enter Allocation Matrix :

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

→

PAGE NO :
DATE :

Enter the max matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter the available Resources:
3 3 2

Following is the Safe Sequence:
P_1, P_3, P_4, P_0, P_2

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

Enter allocation Matrix:
0 2 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter the max matrix:
8 4 6
3 5 7
3 6 7
9 5 3
2 5 7

Enter the available resource: 3 2 2
The following system is not safe

2) Deadlock Detection:

Program:

```
#include<stdio.h>

int max[100][100];
int allocation[100][100];
int need[100][100];
int available[100];
int n,r;

int main()
{
   int i,j;
   printf("Deadlock Detection\n");
   input();
   show();
   cal();
   return 0;
}

void input()
{
   int i,j;
   printf("Enter the no of Processes: ");
   scanf("%d",&n);
   printf("Enter the no of resource instances: ");
   scanf("%d",&r);
   printf("Enter the Max Matrix:\n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<r;j++)
      {
         scanf("%d",&max[i][j]);
      }
   }
   printf("Enter the Allocation Matrix:\n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<r;j++)
      {
         scanf("%d",&allocation[i][j]);
      }
   }
   printf("Enter the available Resources:\n");
   for(j=0;j<r;j++)
   {
      scanf("%d",&available[j]);
   }
}

void show()
{
   int i,j;
   printf("Process\t Allocation\t Max\t Available\t");
   for(i=0;i<n;i++)
```

```
    {
        printf("\nP%d\t   ",i+1);
        for(j=0;j<r;j++)
        {
            printf("%d ",allocation[i][j]);
        }
        printf("\t");
        for(j=0;j<r;j++)
        {
            printf("%d ",max[i][j]);
        }
        printf("\t");
        if(i==0)
        {
            for(j=0;j<r;j++)
            printf("%d ",available[j]);
        }
    }
}

void cal()
{
    int finish[100],temp,need[100][100],flag=1,k,c1=0;
    int dead[100];
    int safe[100];
    int i,j;
    for(i=0;i<n;i++)
    {
        finish[i]=0;
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<r;j++)
        {
            need[i][j]=max[i][j]-allocation[i][j];
        }
    }
    while(flag)
    {
        flag=0;
        for(i=0;i<n;i++)
        {
            int c=0;
            for(j=0;j<r;j++)
            {
                if((finish[i]==0)&&(need[i][j]<=available[j]))
                {
                    c++;
                    if(c==r)
                    {
                        for(k=0;k<r;k++)
                        {
                            available[k]+=allocation[i][j];
                            finish[i]=1;
```

```
                    flag=1;
                }
                if(finish[i]==1)
                {
                    i=n;
                }
            }
        }
    }
}
j=0;
flag=0;
for(i=0;i<n;i++)
{
    if(finish[i]==0)
    {
        dead[j]=i;
        j++;
        flag=1;
    }
}
if(flag==1)
{
    printf("\n\nSystem is in Deadlock and the Deadlock process are\n");
    for(i=0;i<n;i++)
    {
        printf("P%d\t",dead[i]);
    }
}
else
{
    printf("\nNo Deadlock Occur");
}
}
```

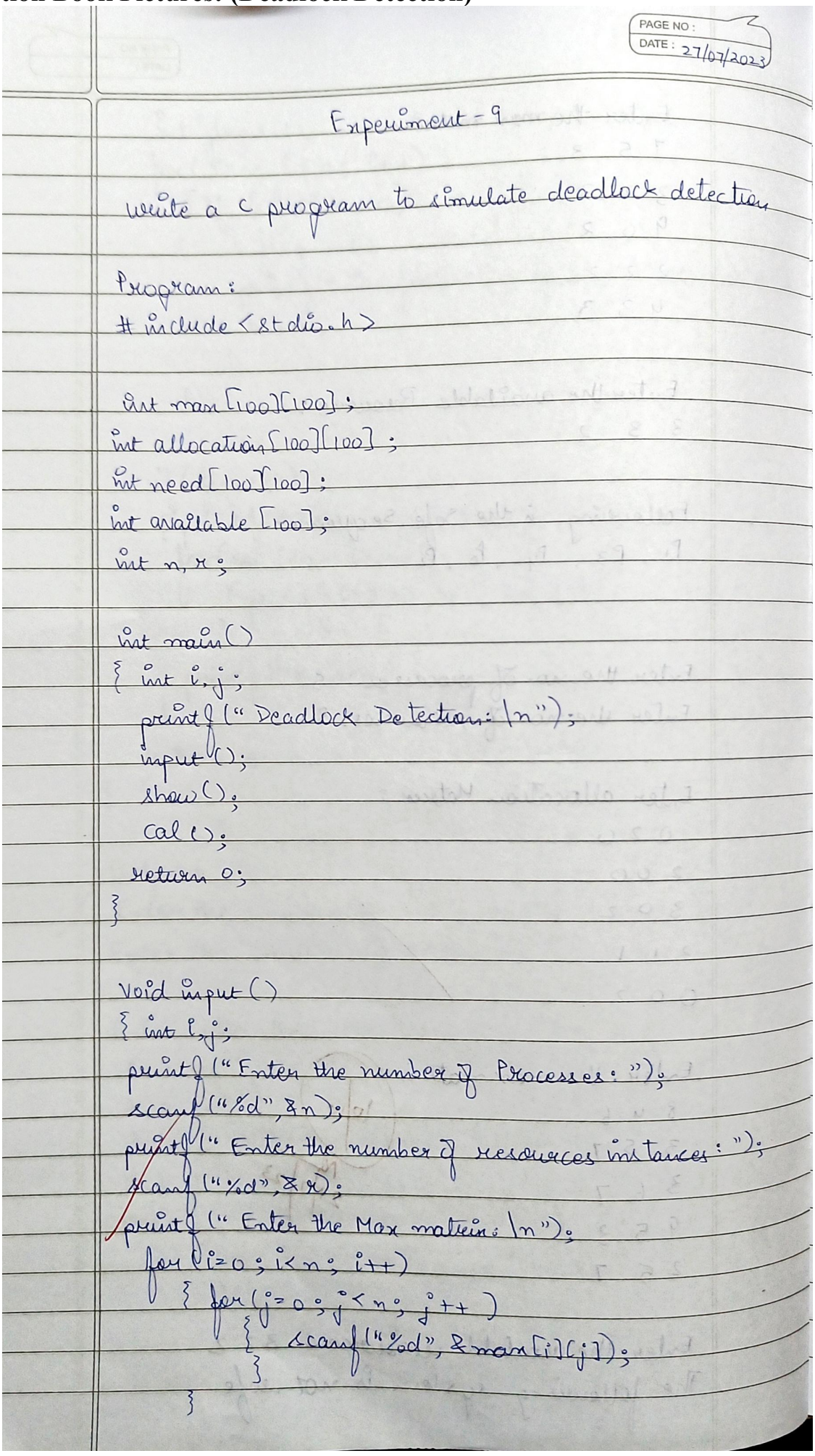
Output: (Deadlock Detection)

```
Deadlock Detection
Enter the no of Processes: 3
Enter the no of resource instances: 3
Enter the Max Matrix:
3 6 8
4 3 3
3 4 4
Enter the Allocation Matrix:
3 3 3
2 0 4
1 2 4
Enter the available Resources:
1 2 0
Process  Allocation      Max     Available
P0         3 3 3        3 6 8   1 2 0
P1         2 0 4        4 3 3
P2         1 2 4        3 4 4

System is in Deadlock and the Deadlock process are
P0      P1      P2
```

```
Deadlock Detection
Enter the no of Processes: 5
Enter the no of resource instances: 3
Enter the Max Matrix:
0 0 0
2 0 2
0 0 0
1 0 0
0 0 2
Enter the Allocation Matrix:
0 1 0
2 0 0
3 0 3
3 1 1
0 0 2
Enter the available Resources:
0 0 0
Process  Allocation      Max     Available
P0         0 1 0        0 0 0   0 0 0
P1         2 0 0        2 0 2
P2         3 0 3        0 0 0
P3         3 1 1        1 0 0
P4         0 0 2        0 0 2
No Deadlock Occur
```

Observation Book Pictures: (Deadlock Detection)



PAGE NO :
DATE : 27/07/2023

Experiment - 9

write a c program to simulate deadlock detection

Program:

```
#include <stdio.h>

int max[100][100];
int allocation[100][100];
int need[100][100];
int available[100];
int n, r;

int main()
{ int i, j;
  printf("Deadlock Detection: \n");
  input();
  show();
  cal();
  return 0;
}

Void input()
{ int i, j;
printf("Enter the number of Processes: ");
scanf("%d", &n);
printf("Enter the number of resources instances: ");
scanf("%d", &r);
printf("Enter the Max matrix: \n");
  for(i=0; i<n; i++)
    { for(j=0; j<n; j++)
        { scanf("%d", &max[i][j]);
        }
    }
```

```
    printf(" Enter the available resources: \n");
    for (j=0; j< r; j++)
        { scanf("%d", &available[j]);
        }
}

void show()
{ int i, j;
  printf(" Process \t Allocation \t Max \t Available \t");
  for(i=0; i< n; i++)
    { printf("\n P%d \t ", i);
      for(j=0; j < r; j++)
        { printf("%d", allocation[i][j]);
        }

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

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

void cal()
{ int finish[100], need[100][100], temp, k, flag=1, c1=0;
  int dead[100];
  int safe[100];
  int i, j;
```

```
for(i=0; i<n; i++)
{ for(j=0; j<r; j++)
  { need[i][j] = max[i][j] - allocation[i][j];
  }
}

while(flag)
{ flag=0;
  for(i=0; i<n; i++)
  { int c=0;
    for(j=0; j<r; j++)
    { if((finish[i]==0) && (need[i][j] <= available[j]))
      { c++;
        if(c==r)
        { for(k=0; k<r; k++)
          { available[k] += allocation[i][j];
            finish[i] = 1;
            flag = 1;
          }
        if(finish[i]==1)
        { i=n;
        }
      }
    }
  }
}
j=0;
flag=0;
```

```
for(i=0; i<n; i++)
{ if(finish[i]==0)
  { dead[j]=i;
    j++;
    flag=1;
  }
}
if(flag==1)
{ printf("\n\n System is in Deadlock and the
          Deadlock process are:\n");
  for(i=0; i<n; i++)
  { printf("%d\t", dead[i]);
  }
}
else
{ printf("\n No Deadlock Occur");
}
}
```

Output:

a) Deadlock Detection:

Enter the number of processes: 3

Enter the number of resource instances: 3

Enter the Max matrix:

3 6 8

4 3 3

3 4 4

Enter the Allocation matrix:

3 3 3

2 0 4

1 2 4

→

DATE :

Enter the available resources :
1 2 0

Process	Allocation	Max	Available
P_0	3 3 3	3 6 8	1 2 0
P_1	2 0 4	4 3 3	
P_2	1 2 4	3 4 4	

System is in Deadlock and the Deadlock process are:
P_0 P_1 P_2

b) Deadlock Detection:
Enter the number of Processes : 5
Enter the number of resource instances : 3
Enter the Max matrix :
0 0 0
2 0 2
0 0 0
1 0 0
0 0 2

Enter the Allocation Matrix :
0 1 0
2 0 0
3 0 3
3 1 1
0 0 2

Enter the available resources :
0 0 0

→

PAGE NO :
DATE :

Process	Allocation	Max	Available
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	3 1 1	1 0 0	
P_4	0 0 2	0 0 2	

10/10

No Deadlock.

28/7/23