**Name**: Akshat Rajesh Andhale **Class**: TEIT

**Roll** **no**.: 03

**Assignment no.: 05** (Deadlock Avoidance Algorithm: Bankers Algorithm)

**CODE:**

#include<stdio.h>

int main()

{

  /\* array will store at most 5 process with 3 resoures if your process or

  resources is greater than 5 and 3 then increase the size of array \*/

  int p, c, count = 0, i, j, alc[5][3], max[5][3], need[5][3], safe[5], available[3], done[5], terminate = 0;

  printf("Enter the number of process and resources");

  scanf("%d %d", & p, & c);

  // p is process and c is diffrent resources

  printf("enter allocation of resource of all process %dx%d matrix", p, c);

  for (i = 0; i < p; i++)

  {

    for (j = 0; j < c; j++)

    {

    scanf("%d", & alc[i][j]);

    }

  }

  printf("enter the max resource process required %dx%d matrix", p, c);

  for (i = 0; i < p; i++)

  {

    for (j = 0; j < c; j++)

    {

      scanf("%d", & max[i][j]);

    }

  }

  printf("enter the  available resource");

  for (i = 0; i < c; i++)

    scanf("%d", & available[i]);

    printf("\n need resources matrix are\n");

    for (i = 0; i < p; i++)

    {

      for (j = 0; j < c; j++)

      {

        need[i][j] = max[i][j] - alc[i][j];

        printf("%d\t", need[i][j]);

      }

      printf("\n");

    }

    /\* once process execute variable done will stop them for again execution \*/

    for (i = 0; i < p; i++)

    {

      done[i] = 0;

    }

    while (count < p)

    {

      for (i = 0; i < p; i++)

      {

        if (done[i] == 0)

        {

          for (j = 0; j < c; j++)

          {

            if (need[i][j] > available[j])

              break;

          }

          //when need matrix is not greater then available matrix then if j==c will true

          if (j == c)

          {

            safe[count] = i;

            done[i] = 1;

            /\* now process get execute release the resources and add them in available resources \*/

            for (j = 0; j < c; j++)

            {

              available[j] += alc[i][j];

            }

            count++;

            terminate = 0;

          }

          else

          {

            terminate++;

          }

        }

      }

      if (terminate == (p - 1))

      {

        printf("safe sequence does not exist");

        break;

      }

    }

    if (terminate != (p - 1))

    {

      printf("\n available resource after completion\n");

      for (i = 0; i < c; i++)

      {

        printf("%d\t", available[i]);

      }

      printf("\n safe sequence is\n");

      for (i = 0; i < p; i++)

      {

        printf("p%d ->\t", safe[i]);

      }

    }

    return 0;

}

**OUTPUT:**

Enter the number of process and resources

5 3

enter allocation of resource of all process 5x3 matrix

0 1 2

3 0 3

2 0 0

2 1 2

0 0 3

enter the max resource process required 5x3 matrix

7 5 3

3 2 2

9 0 2

4 2 2

5 3 3

enter the available resource 3 3 1

need resources matrix are

7 4 1

0 2 -1

7 0 2

2 1 0

5 3 0

available resource after completion

10 5 11

safe sequence is

p1 -> p3 -> p4 -> p0 -> p2 ->

...Program finished with exit code 0

Press ENTER to exit console.