CODE THUẬT TOÁN BANKER

Cho bảng cấp phát tài nguyên của tiến trình

Process	Max			Allocation			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P0	7	5	3	0	1	0	3	3	2
P1	3	2	2	2	0	0			
P2	9	0	2	3	0	2			
P3	2	2	2	2	1	1			
P4	4	3	3	0	0	2			

Yêu cầu:

- Sử dụng giải thuật banker để xác định yêu cầu của P1(1,0,2) cấp phát được không

Code hướng dẫn

```
class Program
    {
        //Ham tinh ma tran need
        static void calculateNeed(int numP, int numR,int[,] need, int[,] maxm, int[,]
allot)
        {
            for (int i = 0; i < numP; i++)
                for (int j = 0; j < numR; j++)
                    // Need of instance = maxm instance - allocated instance
                    need[i, j] = maxm[i, j] - allot[i, j];
        }

        // Ham xac dinh trang thai an toan
        static bool isSafe(int numP, int numR, int[] processes, int[] avail, int[,] need,
                    int[,] allot)
        {
            // Luu trang thai cac tien trinh ket thuc
            bool[] finish = new bool[numP];

            // Luu chuoi cap phat an toan
            int[] safeSeq = new int[numP];

            // Tao ban sao tu avail
            int[] availC = new int[numR];
            for (int i = 0; i < numR; i++)
                availC[i] = avail[i];


            int count = 0;
```

```
            while (count < numP)
            {
                bool found = false;
                //Tim tien trinh chua ket thuc va cap phat duoc
                for (int p = 0; p < numP; p++)
                {
                    //Kiem tra tien trinh chua ke thuc
                    if (finish[p] == false)
                    {
                        //Kiem tra tat ca cac tai nguyen j deu cap phat duoc
                        int j;
                        for (j = 0; j < numR; j++)
                            if (need[p, j] > availC[j])
                                break;

                        if (j == numR)
                        {
                            //Gia su cap hat het va thu hoi tai nguyen
                            for (int k = 0; k < numR; k++)
                                availC[k] += allot[p, k];

                            //Them tien trinh vao chuoi cap phat an toan
                            safeSeq[count++] = p;

                            // Danh dau ket thuc
                            finish[p] = true;

                            found = true;
                            break;
                        }
                    }
                }

                // Truong hop tim khong duoc tien trinh de cap phat
                if (found == false)
                {
                    Console.Write("\nHe thong khong an toan");
                    return false;
                }
            }

            // He thong an toan va in chuoi cap phat an toan
            Console.Write("\nHe thong an toan, thu tu cap phan an toan la:");
            for (int i = 0; i < numP; i++)
                Console.Write(safeSeq[i] + " ");

            return true;
        }
        static bool Banker(int p, int[] Request, int numP, int numR, int[] processes,
int[] avail, int[,] need,
                    int[,] allot)
        {
            //Tao ban sao need, avail, allot
            int[,] needC = new int[numP, numR];
            int[,] allotC = new int[numP, numR];
            int[] availC = new int[numR];
            for(int i = 0; i<numP; i++)
                for(int j=0; j<numR; j++)
```

```
        {
            needC[i, j] = need[i, j];
            allotC[i, j] = allotC[i, j];
        }
    for (int i = 0; i < numR; i++)
        availC[i] = avail[i];

    //Gia su cap phat
    for (int r=0; r<numR; r++)
    {
        //Kiem tra yeu cau hop le va cap phat duoc
        if(need[p, r] >= Request[r]&& avail[r] >= Request[r])
        {
            //GS cap phat
            need[p, r] -= Request[r];
            allot[p, r] += Request[r];
            avail[r] -= Request[r];
        }else
        {
            Console.Write("\n Yeu cau khong hop le!!!");
            return false;
        }
    }
    //Kiem tra he thong an toan
    bool flag = isSafe(numP, numR, processes, avail, need, allot);

    //In ket qua
    Console.Write("\nYeu cau cua P1 : ");
    for (int i = 0; i < numR; i++)
        Console.Write(Request[i] + " ");
    if (flag)
        Console.Write(" cap phat duoc.");
    else
        Console.Write(" phai doi!!!");
    return flag;
}

static void Main(string[] args)
{
    int numP = 5, numR = 3;
    int[] processes = { 0, 1, 2, 3, 4 };

    // Luu the hien tu do cua cac tai nguyen
    int[] avail = { 3, 3, 2 };

    // Luu nhu cau Max cua tat cac ca tien trinh voi cac tai nguyen
    int[,] maxm = { {7, 5, 3},
                    {3, 2, 2},
                    {9, 0, 2},
                    {2, 2, 2},
                    {4, 3, 3}};

    // Luu trang thai da cap phat cua tat ca cac tai nguyen
    int[,] allot = {{0, 1, 0},
                    {2, 0, 0},
                    {3, 0, 2},
                    {2, 1, 1},
                    {0, 0, 2}};
```

```
        // Tinh ma tran need
        int[,] need = new int[numP, numR];
        calculateNeed(numP, numR, need, maxm, allot);

        //Gia Su P1 yeu cau
        int[] Request = { 1, 0, 2 };
        Banker(1, Request, numP, numR, processes, avail, need, allot);
        Console.ReadLine();
    }
}
```