```
int ar[1001][1001];

int kadane(int* arr, int* start, int* finish, int n)
{
    int sum = 0, maxSum = INT_MIN, i;
    *finish = -1;
    int local_start = 0;

    for(i=0;i<n;i++)
    {
        sum += arr[i];
        if (sum < 0)
        {
            sum = 0;
            local_start = i+1;
        }
        else if (sum > maxSum)
        {
            maxSum = sum;
            *start = local_start;
            *finish = i;
        }
    }

    if (*finish != -1)
        return maxSum;

    maxSum = arr[0];
    *start = *finish = 0;

    for (i = 1; i < n; i++)
    {
        if (arr[i] > maxSum)
        {
            maxSum = arr[i];
            *start = *finish = i;
        }
    }
    return maxSum;
}

pair<int,int> findMaxSum(int ROW, int COL)
{
    int maxSum = INT_MIN+1, finalLeft, finalRight, finalTop, finalBottom;

    int left, right, i;
    int temp[ROW], sum, start, finish;

    for (left = 0; left < COL; ++left)
    {
        memset(temp, 0, sizeof(temp));

        for (right = left; right < COL; ++right)
```

```
        {
            for (i = 0; i < ROW; ++i)
                temp[i] += ar[i][right];
            sum = kadane(temp, &start, &finish, ROW);

            if (sum > maxSum)
            {
                maxSum = sum;
                finalLeft = left;
                finalRight = right;
                finalTop = start;
                finalBottom = finish;
            }
        }
    }
    int p = finalRight - finalLeft, q = finalBottom - finalTop;
    return {p+1,q+1};
}
```