**P1. M09043BB**

**3.**

**(a) What is dynamic programming?**

**Dynamic programming** is typically applied to optimization problems. In such problem there can be many solutions. Each solution has a value, and we wish to find a solution with the optimal value.

**The development of a dynamic programming algorithm** can be broken into a sequence of four steps:

**1.** Characterize the structure of an optimal solution.

**2.** Recursively define the value of an optimal solution.

**3.** Compute the value of an optimal solution in a bottom up fashion.

**4.** Construct an optimal solution from computed information.

**(b) Write a dynamic-programming algorithm to calculate C (n, k), the number of k-combinations (i.e., k-element subsets) of an n-element set. Use the formulas:**

C (n, k) = C (n - 1, k - 1) + C (n - 1, k)

valid for 1 <= k <= n - 1, and C (n, n) = 1 = C (n, 0) valid for n >= 0.



Combine(n, k)

for i = 0 to n do

C[i, i] = 1

C[i, 0] = 1

for i = 2 to n do

for j = 1 to n – 1 do

C[i, j] = C[i – 1, j – 1] + C[i – 1, j]

return C[n, k]

Note that C[n, k] is a symmetric matrix, so we can modify code as follows:

Combine(n, k)

m = k

if (m > n / 2) then m= n / 2

for i = 0 to n do

C[i, 0] = 1

for i = 0 to m do

C[i, i] = 1

for i = 2 to n do

for j = 1 to n – 1 do

if (j <= m)

C[i, j] = C[i – 1, j – 1] + C[i – 1, j]

if (k > n – k) then

return C[n, n – k]

return C[n, k]

**(c) The worst-case time of this algorithm is O (n2)**

**(d) Trace your algorithm for C (7, 5):**

