You must show your work for full credit. Guessing or simply “knowing” the answer will only be worth partial credit. You may make assumptions to remove the floor and ceiling functions. You may not be able to show all your work on the actual exam so write your name and student number on each sheet you turn in. You are allowed one sheet of notes (front and back). You do not need to turn in your notes sheet.

1) In Build-MaxHeap(A),

Protocol 1 Build-MaxHeap(A)

1: procedure Build-MaxHeap(A)
2: heap-size = length[A];
3: for i=floor(heap-size/2) downto 1 do
4: do Max-Heapify(A,i)
5: end for
6: end procedure

a) (10pts) Give the formal pre-conditions for the algorithm Build-MaxHeap(A)
b) (10pts) Develop a loop invariant for this loop leading to the correct post condition that A is a max-heap.

2) Your job is to design an algorithm to dispense change using the fewest possible coins.
   a) (15pts) Given the US coin system (Q=25, D=10, N=5, P=1) write an algorithm to make change in coins for any amount less than one dollar (one dollar = 100).
   b) (5pts) Is it possible to modify your algorithm for any coinage system? Explain or give a counter example.
3) You have the following algorithm

**Protocol 2 LCS-Length(A)**

1: procedure LCS-LENGTH(X,Y)  
2:   \[m = X.length;\]  
3:   \[n = Y.length;\]  
4:   let \(c[0 \ldots m, 0 \ldots n]\) be a new table  
5:   for \(i = 0\) to \(m\) do  
6:     \(c[i, 0] = 0\)  
7:   end for  
8:   for \(j = 0\) to \(n\) do  
9:     \(c[0, j] = 0\)  
10:   end for  
11:   for \(i = 1\) to \(m\) do  
12:     for \(j = 1\) to \(n\) do  
13:       if \(x_i == y_j\) then  
14:         \(c[i, j] = c[i - 1, j - 1] + 1\)  
15:       else if \(c[i - 1, j] \geq c[i, j - 1]\) then  
16:         \(c[i, j] = c[i - 1, j]\)  
17:       else  
18:         \(c[i, j] = c[i, j - 1]\)  
19:       end if  
20:     end for  
21:   end for  
22: return \(c[m, n]\)  
23: end procedure

a) (15pts) Given LCS-Length(X,Y) show the correctly how the procedure finds the LCS for \(X=“CCAGTAC”\) and \(Y=“CATG’\).

b) (5pts) What does this algorithm return as the LCS?

4) The next two questions relate to the 0/1 Knapsack Problem with a capacity of 30 and the following weights and values: \(w_i = [40, 15, 15]\) and \(p_i = [30, 25, 25]\)

a) Find the Optimal Solution using dynamic programming

b) Find a solution using a greedy algorithm

c) Which is the best solution given these inputs?

5) Briefly define the Principle of Optimality