NOTE! Problem 1 is given both in pseudocode (page 1) and in C (page 2). Choose whichever one you wish.

(1, 20%) (Pseudocode) Given the following array of numbers and algorithm, answer the questions below. Assume that the global array **T[1..n]** is correctly declared and contains the values shown.

Array T	**2**	**7**	**3**	**8**	**4**	**7**	**5**	**1**
position	1	2	3	4	5	6	7	8

```
procedure R(a : integer)
  i, j, x, y, z : integer;

   x ← 0;
  y ← 0;
  z ← 0;
  for i ← 1 to n do
      for j ← (i+1) to n do
          if (T[i] > T[j]) then
              if (z < a) then
                  z ← z + j;
              endif
              y ← T[i];
              T[i] ← T[j];
              T[j] ← y;
              x ← x + 1;
          endif
      endfor
  endfor
endprocedure
```

a) Show the array **T** after the procedure was called with **R(10)**?

Array T	**1**	**2**	**3**	**4**	**5**	**7**	**7**	**8**
position	1	2	3	4	5	6	7	8

This is a sorting algorithm arranging the values in ascending order by array position. A[I]<=A[j] if and only if I <= J

b) What value will the following variables contain right before the procedure terminates?

x	**12**	**z**	**11**

c) What is the purpose of variable x (in general) while the procedure R is executing?

The variable x counts the number of times the values of A[i] and A[j] are swapped or exchanged.

(1, 20%) **(C code)** Given the following array of numbers and algorithm, answer the questions below. Assume that the integer array **T[1..n]** of size n is correctly declared and contains the values shown.

Array T	2	7	3	8	4	7	5	1
position	0	1	2	3	4	5	6	7

```
void R(int a) {
  int i, j, x=0, y=0, z=0;

  for (i=0; i<n; i++) {
      for (j=i+1; i<n; j++) {
          if (T[i] > T[j]) {
              if (z < a) {
                  z = z + j;
              }
              y = T[i];
              T[i] = T[j];
              T[j] = y;
              x = x + 1;
          }
      }
  }
}
```

a) Show the array **T** after the procedure was called with **R(10)**?

Array T	1	2	3	4	5	7	7	8
position	0	1	2	3	4	5	6	7

This is a sorting algorithm arranging the values in ascending order by array position.
A[I]<=A[j] if and only if I <= J

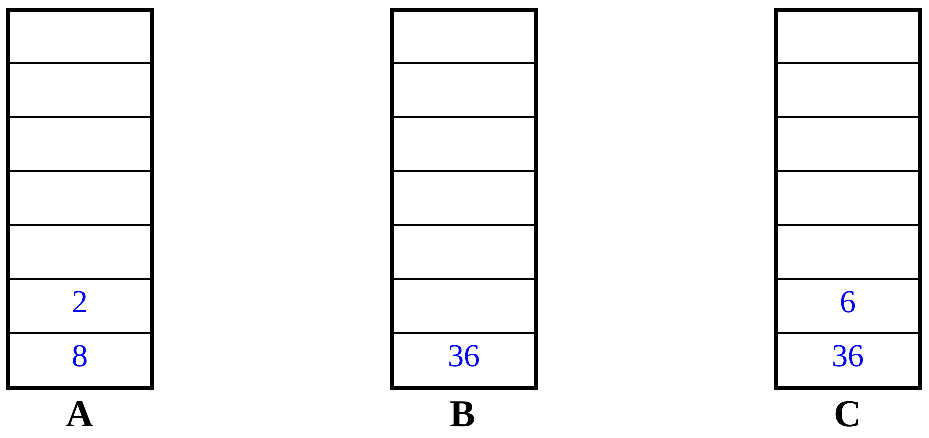
b) What value will the following variables contain right before the procedure terminates?

x	12	z	16

c) What is the purpose of variable x (in general) while the procedure R is executing?
The variable x counts the number of times the values of A[i] and A[j] are swapped or exchanged.

(2, 18%) The following is a Postfix expression. All values are single decimal digits and the operations are addition "+", subtraction "–", multiplication "*" and division "/". (Note that the final answer and intermediate answers in the stack may not be single decimal digits.) In each box below the Postfix expression, show ONLY the contents of the stack at the indicated point in the Postfix string (point A, B or C). Put the final answer in the blank. If the Postfix string is invalid, carry the operations as far as possible and write "invalid" as the answer. (6 points)

a) **8 7 5 –** $^{\mathbf{A}}$ **/ 9 *** $^{\mathbf{B}}$ **9 3 / 2 *** $^{\mathbf{C}}$ **–** = **30**



b) Consider implementing a stack using a linked list. If you push a value on top of the stack by inserting a node storing that value onto the front of the linked list, how would a pop be implemented? Why?

A stack is a Last In First Out (LIFO) memory device. Therefore, if a push adds an element to the front of the list, then a pop must remove that element by deleting the front of the list.

c) Imagine having two stacks, A and B with A containing n values and B containing {}. Assume that the only standard functions available to operate on each of these stacks are push, pop and empty. What is the minimum total number of pops and pushes necessary to remove the bottom element from stack A and return stack A to contain all the elements it did previously except for the removed element? **Total #pops + #pushes = 4N - 3**

```
while ( not Empty(A) {
      x ← Pop(A);                        // N pops
      if not Empty(A) then Push(B,x);  // N-1 pushes
}
while ( not Empty(B) {
       Push(A,Pop(B))                     // N-1 pops and N-1 pushes
}
```

(3, 20%) Answer each of the following "timing" questions concerning an algorithm of a particular order and a data instance of a particular size. Assume that the run time is affected by the size of the data instance and not its composition. (Assume that $\lg_2 n = \log_2 n$.)

a) (4 pts) For an **O($\lg_2$n)** algorithm, an instance with n=32 takes 15 milliseconds to run. How long would it take to run the algorithm with n=128? **21**

$$\frac{\log_2(32)}{15} = \frac{\log_2(128)}{x}$$

$$x = \frac{15\times\log_2(128)}{\log_2(32)} = \frac{15\times 7}{5} = 21$$

b) (4 pts) For an **O(Ön)** algorithm, an instance with **n = 25** takes **70** milliseconds. If you used a different-sized data instance and it took **42** milliseconds how large must that instance be? **n = 9**

$$\frac{\sqrt{25}}{70} = \frac{\sqrt{n}}{42}$$

$$\frac{42\times 5}{70} = \sqrt{n}$$

$$\frac{210}{70} = 3 = \sqrt{n}$$

$$9 = n$$

c) (4 pts) Consider 2 algorithms A and B, both used to solve the identical problem. Algorithm A is an **O(n^2)** algorithm whereas algorithm B is an **O(n^3)** algorithm. It turns out that for input size n=10, the algorithms have identical running times. For what value of n does algorithm A run four times as fast as algorithm B?

$\text{running time for } A = A(n) = an^2, \text{ for some constant } a$

$\text{running time for } B = B(n) = bn^3, \text{ for some constant } b$

$\text{For } n = 10,\ A(n) = B(n) \text{ implies } 100a = 1000b. \text{ Therefore } \frac{a}{b} = 10.$

$\text{Now consider for what value of } n \text{ is it true that } B(n) = 4A(n)?$

$\text{Substituting for } A(n) \text{ and } B(n) \text{ we have: } bn^3 = 4an^2,\ n = \frac{4a}{b} = 4(\frac{a}{b}) = 4(10) = 40$

(3 continued)
Given the following algorithm, answer the questions below for an arbitrary positive integer **n**.
Once again, the algorithm is presented in both pseudocode and C code. Choose which one you prefer.

Pseudocode Version:

```
x ← 0
for i ← 1 to n do
  if (x < i)
       for j ← 1 to 2*n do
            x ← x + j
       end for
  end if
end for
```

C code Version:

```
x = 0;
for( i = 1; i <= n; i++ ) {
    if ( x < i ) {
       for( j = 1; j <= 2*n; j++ )  x = x + j;
    }
 }
```

d) (2 pts) What is the Order of this code segment, in terms of n? ***O(n)***

This follows because the inner for-loop can only execute once (when x = 0 and i = 1). The outer loop is O(n), and the inner loop is O(n).

e) (6 pts) What will be the value of **x** (in terms of n) when the **for** loops end?

$$x = \sum_{j=1}^{2n} j = \frac{2n(2n+1)}{2} = 2n^2 + n$$