CSC 1051: Algorithms and Data Structures I – Sample Algorithm Questions

Construct an algorithm that inputs a number num and prints all its factors (i.e., all the values of x in the range 2…(num-1) such that num is divisible by x). After the factors are printed, print a goodbye message. Note: You are NOT being asked to print the only prime factors (that one is more challenging).

Example: If num (i.e., the input) is 20, the algorithm should print:

The factors of 20 are:
2  4  5  10
Goodbye

Directions:
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```
input num
input x
x = 1
x = 2
x = x + 1
if (num % x == 0)
if (num % 2 == 0)
else
while (x < num)
while (x <= num)
while (num % 2 == 0)
print "The factors of " num " are:"
print x
print num
print "Goodbye"
```

Construct an algorithm that inputs some numbers from the user, terminated with a zero (sentinel value). The algorithm should print the numbers entered and compute and print their sum. After the sum is printed, print a goodbye message.

Example: If the numbers 5 -3 8 0 are entered as input, the algorithm should print:

5
-3
8
Sum = 10
Goodbye

[Note that the terminating zero should NOT be printed.]

Directions:
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```
input num
input sum
num = 0
sum = 0
num = num + 1
sum = sum + 1
sum = sum + num
sum = num + 1
if (sum == num)
if (sum < num)
if (num < sum)
else
while (num < sum)
while (num < sum)
while (num != 0)
while (num != sum)
print "The factors of " num " are:"
print "Sum = " sum
print num
print "Goodbye"
```
Construct an algorithm that inputs an integer num and 10,000 other integers and prints a message indicating whether num was found among the 10,000 other integers, followed by a goodbye message. 

Hint: You need to use a boolean variable found to keep track of whether you found a match.

**Example:** If num (i.e., the first number input) is 1318, and 10,000 other numbers are input after that, none of which is equal to 1318, the algorithm should print:

```
Searching for 1318
Not found
Goodbye
```

Alternatively, if the number 1318 occurred one or more times among the other numbers, it should print:

```
Searching for 1318
Found it!
Goodbye
```

**Directions:**
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```plaintext
found = true
found = false
input num
input x
count = 1
count = count + 1
x = x + 1
if (x == num)
if (found)
if (found == true)
if (found == false)
```

---

Construct an algorithm that inputs a number num and then prints "Hello" that many times. After the “Hello”s are printed, print a goodbye message.

**Example:** If num (i.e., the input) is 5, the algorithm should print something like this:

```
Hello
Hello
Hello
Hello
Hello
Goodbye
```

**Directions:**
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```plaintext
input num
input count
count = 1
count = 0
count = count + 1
num = num + 1
if (count < num)
else
while (count <= 10000)
while (num <= 10000)
while (count <= num)
print “Searching for “ num
print x
print num
print “Found it!”
print “Not found”
print “Goodbye”
```

---
Construct an algorithm that inputs 10 positive numbers and prints the maximum of these numbers.

**Example:** If the numbers input are 44 7 31 22 53 16 21 48 72 60, the output should be:
max = 72

**Directions:**
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```plaintext
num = 0
count = 0
count = 1
max = 0
count = count + 1
num = max
max = num
print "max = " num
input num
input max
input count
if (num > max)
else
while (count <= 10)
while (count < 10)
while (num != max)
```

**Construct an algorithm to solve the following problem:**
A detachment of n soldiers must cross a wide and deep river with no bridge in sight. They notice two children playing in a rowboat by the shore. The boat is so tiny, however, that it can only hold two children or one soldier. How can the children help the soldiers get across the river (and themselves wind up back where they started with their boat)? Given a number n of soldiers, specify the steps in solving this problem and determine also how many times the boat has to cross each way.

**Example:** If input n = 5, then the algorithm should produce output that looks like a sequence of moves that will get five soldiers across and leave the children back at the original shore with their boat. Then it should print the number of times the boat crossed the river. Something that looks like this:

```plaintext
one soldier across
one soldier back
two children across
one child back
one soldier across
...
number of crossings 28
```
(This answer is incorrect – showing the right answer would ruin the puzzle!).

**Directions:**
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```plaintext
print "soldier across"
print "soldier back"
print "one child across"
print "one child back"
print "two children across"
print "two children back"
print "soldier back"
print "number of crossings = " count
input n
n = n-1
count = 0
count = 1
count = count + 1
while (n>0)
while (n>=1)
while (count<n)
```
Construct an algorithm that inputs several positive integers in the range 1...100 from the user, terminated with a -1 (sentinel value). The algorithm should print the numbers entered and determine and print the minimum value. After the minimum is printed, print a goodbye message. You can assume that the numbers entered will be between 1 and 100 (except for the terminating -1), so you do not need to check for mistakes in the input.

Note: Be careful not to process the terminating -1, i.e., make sure your algorithm does not produce -1 as the minimum.

Example: If the numbers 25 86 13 54 -1 are entered as input, the algorithm should print:

```
25
86
13
54
Min = 13
Goodbye
```

[Note that the terminating -1 should NOT be printed.]

Directions:
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary:

- input num
- input min
- num = 0
- min = 0
- min = 100
- num = 100
- num = num + 1
- min= num
- num = min
- if (num < min)
- if (num > min)
- if (num != -1)
- if (num != min)
- else
- while (num < min)
- while (num > min)
- while (num != -1)
- while (num != min)
- print "Min = " min
- print "Min = " num
- print num
- print "Goodbye"
Construct an algorithm that inputs 10 positive integers from the user and determines how many of the numbers entered are even (i.e., divisible by 2). The algorithm should print the numbers as they are entered, and meanwhile keep track of how many have been even (hint: use the variable evenCount). After all the numbers have been entered, print the result (how many were even) and a goodbye message.

Example: If the numbers 25 86 13 54 1 52 4 88 9 41 are entered as input, the algorithm should print:

```
25
86
13
55
1
52
9
88
9
41
3 of these numbers are even
Goodbye
```

Directions:
Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary:

```
num = 0
count = 0
count = 1
evenCount= 0
count = count + 1
evenCount = evenCount + 1
num = num + 1
input num
input evenCount
input count
if (num %2 == evenCount)
if (num % 2 == 0)
```