

CS 101 Computer Science I (Spring 2001) Assignment 5

1. Write a Scheme procedure called "**upto**". This procedure will take a positive integer called "**n**" as input. It will return a list of all the positive integers up to and including **n**. The returned list should be ordered so that low numbers come before high numbers. Use the accumulator method in your solution.

```
(upto 1)           ==> (1)
(upto 5)           ==> (1 2 3 4 5)
```

2. Write a Scheme procedure called "**sum-squares**". This procedure will take a positive integer called "**n**" as input. It will return the sum of the squares of all the positive integers up to and including **n**. Use the accumulator method in your solution.

```
(sum-squares 1)   ==> 1
(sum-squares 5)   ==> 55
```

3. Write a Scheme procedure called "**largest**". This procedure will take a list of numbers as input. The numbers on the list may be positive, negative or zero. If the input list is empty, the procedure returns the symbol **none** indicating that there is no largest number on the input list. Otherwise, the procedure returns the largest number on the input list. Use the accumulator method in your solution.

```
(largest '())      ==> none
(largest '(7))     ==> 7
(largest '(6 2 9 1 3)) ==> 9
(largest '(-3 -2 -7)) ==> -2
```

4. Write a Scheme procedure called "**lists-below-above**". This procedure will take as input a list of numbers called "**lst**" and a particular number called "**bound**" as parameters. The procedure will return a list of two lists. Let the two returned lists be called "**list-below**" and "**list-above**" respectively. The **list-below** includes the numbers on **lst** that are less than or equal to **bound**. The **list-above** includes the numbers on **lst** that are greater than **bound**. The order of the members of **list-below** and **list-above** does not matter; however, each number appearing on **list-below** or **list-above** should appear the same number of times as it occurs in the input **lst**. Use the accumulator method in your solution.

```
(lists-below-above '() 7)           ==> (())
(lists-below-above '(1 2 3 4 5 6 7 8 9) 5) ==> ((1 2 3 4 5)(6 7 8 9))
(lists-below-above '(3 1 9 7 4 2 2) 6) ==> ((3 1 4 2 2)(9 7))
```

Due Dates

- Section 51 (Professor Welty): Monday, February 26, 2001
- Section 52 (Professor Ellman): Tuesday, February 27, 2001