Steven Murr

HW 3.1

Problems = { 1, 3, 6, 11, 13, 14, 16 }

1) List all the steps used by Algorithm 1 to find the maximum of the list 1,8,12,9,11,2,14,5,10,4.

I'm going to write Algorithm 1 is pseudo code and comment each line discussing what is happening in each step.

procedure $max(a_1, a_2...a_n : integers)$ Declare a procedure or function called max which takes integers as input
$max := a_1$ the variable max gets the value of a_1
for i := 2 to n iterate from the second position in the list to the length of the list
    if $max < a_i$ **then** max := a_i if the value in max is less then the next iterated value - then put this new value into max.
return max return element in the variable max.

*See attached paper for handwritten table breakdown.

3) Devise an algorithm that finds the sum of all integers in a list.

procedure $sum(a_1, a_2...a_n : integers)$ Declare a function sum which takes integers as input
$sum := a_1$ the variable sum gets the value of a_1
for i := 2 to n iterate from the second position in the list to the length of the list
    sum := sum + a_i the variable sum gets the value in sum + the next value in the list
return sum return the variable sum

6) Describe an algorithm that takes as input a list of n integers and finds the number of negative integers in the list.

procedure $negnums(a_1, a_2...a_n : integers)$ Declare a function/procedure negnums which takes integers as input
$negnums := 0$ initialize a variable called negnums with the value 0 in it - this will be our counter
for i := 1 to n iterate from the first position in the list to the length of the list
    if $a_i < 0$ **then** negnums := negnums + 1 the variable sum gets the value in sum + the next value in the list
return negnums return the variable negnums to the functions caller

11) Describe an algorithm that interchanges the values of the variables x and y, using only assignments. What is the minimum number of assignment statements needed to do this?

procedure $swap(x : anytype, y : anytype)$ Declare a function/procedure swap which takes any two variables as input
$temp := x$ initialize a variable called temp and store the value of x in it.
$x := y$ put the value of y in x.
$y := temp$ put the value of temp in y.

It takes at least three steps to swap two variables.

14) List all the steps used to search for 7 in a sequence given in exercise 13 for both a linear search and a binary search.

procedure $linearsearch(x : integer, a_1, a_2...a_n : distinctintegers)$
$i := 1$ initialize the variable i to the value 1 - this will be used to iterate over the list
while$(i \leq n$ and $x \neq a_i)$continue the following loop while the value of i is less than the length of the set AND the integer we're looking for is not equal to the current item being iterated over
    i := i + 1
if $i \leq n$ **then** location := i if i is less than or equal to n then the variable location gets the value of i
else location := 0 else location gets the value of 0
return location return the variable location to the functions caller

*See attached paper for handwritten table breakdown

14 - Binary Search)

procedure $binarysearch(x : integer, a_1, a_2...a_n : increasingintegers)$
$i := 1$ initialize the variable i to the value 1 - this represents the first element of the list
$j := n$ initialize the variable j to the value n - the length of the list - this represents the last element of the list
while$(i < j)$execute the following loop while i is less then j
  m := $\lfloor (i + j)/2 \rfloor$ take the floor function of i + j divided by 2. This splits the entire set in half
  if x > a_m **then** i := m + 1 If the value we're looking for is greater then the mid point of the list then the smallest bound of the new list is the mid point + 1
  else j := m if it isn't greater then the midpoint, then the max point of the new list is m, which was the midpoint after the split.
if x = a_i**then** location := i if the value we're looking for equals the last value found after splitting the list to conclusion, then puts the value of the index into location - the index is the location in which the variable occurs in the list.
else location := 0 else the value wasn't found and location gets the value 0 to signify the value wasn't found
return location return the variable location to the functions caller

*See attached paper for handwritten table breakdown

16) Describe an algorithm for finding the smallest integer in a finite sequence of natural numbers.

procedure smallest integer($a_1, a_2...a_n$: natural numbers) Declare a procedure/function called smallest integer that takes in a finite list of natural numbers
least := a_1 least gets the first value in the list
for i := 2 **to** n set the variable i to 2 and iterate up to the nth element in the list.
  if least > a_i **then** *least* := a_i if the value is least is greater than the currently iterated item, then least gets the value of the currently iterated item.
return *least* return the variable least to the functions caller

*I chose a linear search algorithm because we don't know if the list is ordered or not. A list must be ordered to use binary search.