Homework Five

CS323 Programming Languages

**Chapter Nine Problems:**

1. Consider the following program written in C++ syntax:

int main(void)

{

 int value = 2, list[5] = {1, 3, 5, 7, 9};

 swap(value, list[0]);

 swap(list[0], list[1]);

 swap(value, list[value]);

}

void swap (int a, int b)

{

 int temp;

 temp = a;

 a = b;

 b = temp;

}

For each of the following parameter passing methods, what are all of the values of the variables value and list after each of the three calls to swap?

* 1. Passed by value
	2. Passed by reference
	3. Passed by value result
	4. Passed by name

**Chapter Ten Problems:**

1. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume Bigsub is at level 0.
2. Show the chain offset and local offset for each variable in the expression at position 1 assuming static scope.
3. Show the variable stack configuration at position 1 assuming dynamic scope and a shallow access implementation.

**procedure** Bigsub **is**

Calling sequence for this pgoram for execution to reach D is:

Bigsub calls A

A calls B

B calls A

A calls C

C calls D

 **procedure** A(Flag : Boolean) **is**

 v,w : Integer

 **procedure** B **is**

u,y : Integer

 ...

 A(false);

 **end**; -- of B

 **if** flag

 **then** B;

 **else** C;

 ...

 **end**; -- of A

 **procedure** C **is**

 u,w,y : Integer;

 **procedure** D **is**

t,z : Integer

 ...

Position 1

 z = t\*u + y-w;

 ...

 **end**; -- of D

...

 D;

 **end**; -- of C

**begin**

t,u,v : Integer

 ...

 A(**true**);

 ...

**end**; -- of BigSub

**Fortran Problems:**

1. What parameter passing method(s) does Fortran 77 use for an integer? for an array of integers?
2. Are local variables in FORTRAN static or stack dynamic? Are local variables that are INITIALIZED to a default value static or stack dynamic? Show me some code with an explanation to back up your answer. Hint: The easiest way to check this is to have your program test the history sensitivity of a subprogram. Look at what happens when you initialize the local variable to a value and when you don’t. You may need to call more than one subprogram to lock in your answer with confidence.
3. Implement a linked list data structure for strings. Include at a minimum the following subroutines:

///////////////////////////////////////////////////////

 // Position Methods

 ///////////////////////////////////////////////////////

 isEnd()

 // Precondition: none

 // Postcondition: if current is NULL because it is past the last element in list

 // returns true

 //

 // otherwise, returns false

 next()

 // Precondition: none

 // Postcondition: if isCurrentValid() == true, current is moved to next node

 // and true is returned.

 //

 // otherwise, current remains unchanged and false is returned

 setPos(pos)

 // Precondition: there are (pos+1) nodes in the list

 // Postcondition: if there are (pos+1) nodes in list, moves current to (pos)th node

 // and returns true

 //

 // otherwise, current remains unchanged, and false is returned

 reset()

 // Precondition: none

 // Postcondition: if the list is not empty, sets current to the first node in list

 //

 // otherwise, current remains unchanged

 isCurrentValid()

 // Precondition: none

 // Postcondition: if current is pointing to a valid list node, returns true

 //

 // otherwise, returns false because current is NULL. This happens

 // when the list is EMPTY, or current is at the end of the list

 ///////////////////////////////////////////////////////

 // Node Methods

 ///////////////////////////////////////////////////////

 insert(value)

 // Precondition: value is a list item to be inserted

 // Postcondition: value is added to the list AFTER the current item

 // current now points to newly added item.

 //

 // if the list is empty then:

 // - a new node is created, and set as the first node in list

 // - current points to the new node

 remove()

 // Precondition: none

 // Postcondition: if (isCurrentValid==true) and this is NOT the ONLY node in list

 // - removes the current node from the list

 // - sets current to point to the node before the deleted node

 // - returns true

 //

 // if (isCurrentValid==true) and this is the ONLY node in list then:

 // - current node is removed from list

 // - current is now invalid (isCurrentValid == false)

 // - returns true

 //

 // if (isCurrentValid==false) then:

 // - no action performed

 // - returns false

 //

 find(value)

 // Precondition: none

 // Postcondition: if value is in list, current pointer moved to first occurrence

 // of value in list and true is returned.

 //

 // otherwise, current remains unchanged, and false is returned

 setCurrentItem(value)

 // Precondition: isCurrentValid() == true

 // Postcondition: if isCurrentValid() == true, changes value stored in current node

 // and returns true

 //

 // otherwise, current remains unchanged, and false is returned

 getCurrentItem()

 // Precondition: isCurrentValid() == true

 // Postcondition: if isCurrentValid() == true, returns the value in current node

 // otherwise, return value is unpredictable

 ///////////////////////////////////////////////////////

 // Global List Manipulation Methods

 ///////////////////////////////////////////////////////

 getLength()

 // Precondition: none

 // Postcondition: returns number of nodes in list, list is unchanged

 isEmpty()

 // Precondition: none

 // Postcondition: true if list is empty, false otherwise

 erase()

 // Precondition: none

 // Postcondition: remove all elements from list, on return list is empty, and

 // isCurrentValid==false

output()

 // Precondition: none

 // Postcondition prints data stored in each node, does not alter list

and execute the following main program:

program solutions

insert(“Bob”)

insert(“Mike”)

insert(“Tim”)

insert(“Kathleen”)

insert(“Becky”)

sort()

output()

write(\*,\*) find(“Bob”)

reset()

remove()

remove()

output()

erase()

output()

end

Place your solutions for 2,3 and 4 in a single file called “hw5.f”. Attach a hard copy of your code and a screen snapshot of the execution results to the homework assignment you hand in class.

**Perl Problems**

1. What parameter passing method(s) does Perl use for a scalar? for an array of scalars?
2. Write a Perl subroutine, changeLiteral, that takes a literal as a parameter to the subroutine. Have the subroutine try to change the parameter. Given the overall design philosophy of Perl, explain the results.

NOTE: Be careful not to use local or global variables in the subroutine... these won’t actually change

the literal! Note a literal is a string constant like the string “this is a string constant”.

1. What data structure does Perl use to organize parameters passed to a subroutine? Write a perl subroutine myTenStrings which takes ten strings as arguments, and outputs the strings.
2. Write a Perl subroutine printAndSwap, that takes two arrays and an index as parameters and does the following:
	* prints the value at the index position in the first array
	* prints the value at the index position in the second array
	* prints the value of the index
	* swaps the value at the index position in the first array with the value at the index position in the second array.

 To solve this problem:

* You cannot use global variables inside printAndSwap
* You cannot pass more than three parameters inside printAndSwap

The solution to this problem is not as straightforward as it seems. To get you going in the right direction, FIRST implement the subroutine printArray, then move onto printAndSwap.

Here’s a main program to get you started:

@first = (1, 2, 3, 4);

@second = (10, 20, 30, 40);

print "Before call\n";

print " First:";

&printArray(@first);

print " Second:";

&printArray(@second);

&printAndSwap(...);

print "After call\n";

print " First:";

&printArray(@first);

print " Second:";

&printArray(@second);

Here’s the expected output from the program (assuming we swap at index 1 in both arrays):

Before call

 First:1 2 3 4

 Second:10 20 30 40

Value to swap in first array: 2

Value to swap in second array: 20

Index to swap at: 1

After call

 First:1 20 3 4

 Second:10 2 30 40

Place your solutions for 2,3, and 4 in a single file called “hw5.pl”. Attach a hard copy of your code and a screen snapshot of the execution results to the homework assignment you hand in class.