COMP 273

Assignment 3
School of Computer Science
McGill University
Available On: November 8th, 2017
Due Date: November 22, 2017. 11:59pm.
Submit your solution in electronic form using MyCourses
Read the submission instructions in the end of the document
(late policy: 25 marks off per day late)
This assignment is meant to be done alone.
Sharing code is strictly prohibited.

1 Dynamic Memory Allocation and Linked Lists

For this question, your job is to implement a SINGLY Linked list. This linked list will be built by taking SINGLE CHARACTER inputs from the user, one input at a time, and placing the input in respective nodes. Every node will have a character specified by user input AND an address. The address will be pointing toward the next node in the linked list. You will then develop a procedure to print the linked list you have made, along with a procedure to reverse said linked list. Use whichever subroutines you like. Follow convention.

(a) Write a procedure called "build" that will continually ask the user for another input UNTIL the user has input the character "*". "*" Will act as the token that signifies that the list is complete. The final "*" will not be considered a part of the linked list. For EACH user input until "*", you will put that character into a single node in the linked list. $v1 will be used to return the address of the first node in the linked list.

(b) Write a procedure called "print" which will use $a0 to take the address of the first node of a linked list. "print" will print the contents of each node as it goes through the linked list.

(c) Write a procedure called "reverse" which will reverse a singly linked list. Reverse will take the first node of a linked list as an argument. Pass the address of the first node of a linked list through $a1. Reverse will then reverse every element in the linked list. Return the reversed linked list in $v1. Use whichever subroutines you like. Follow convention.

2 Dr. Ackermann or: How I Learned to Stop Worrying and Love Recursion

There was some confusion on the last assignment about what recursion really entailed. In particular, because Dijstrka's algorithm was "tail-recursive," one could implement it using a single stack frame using a loop, without the need to make a recursive call at all. Functions that can be computed in such a way are called "primitive recursive."

One may ask if all computable functions are primitive recursive. In 1928, German mathematician William Ackermann answered the question in the negative by providing an example of a recursive function which was not primitive recursive (actually, David Hilbert provided the example, but Ackermann formally proved the result). In a sense, the recursion cannot be "taken out" of the computation of the function. In particular, the function cannot be computed using a loop whose bound is known at runtime, so that we cannot get away with using a single stack frame.

The Ackermann function $A(m, n)$ for natural numbers m, n is defined by

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases} \quad (1)$$

Your task is to write a MIPS program that computes the Ackermann function. It should take input from the console and print the answer. You should write a helper procedure that checks the type and range of the inputs, printing an appropriate error message if necessary. You should use the table of values found at https://en.wikipedia.org/wiki/Ackermann_function#Table_of_values to test your program. Be careful: $A(m, n)$ grows *very* rapidly. Thus, you should limit your tests to values of $m < 4$ and $n < 5$.

The history of Ackermann's discovery is a fascinating read. If you are fascinated by the relevant concepts, you should consider taking COMP 330 (Theory of Computation) and COMP 302 (Programming Languages and Paradigms) to learn more.

3 Numerical Integration with the Floating Point Coprocessor

MIPS has two coprocessors dedicated to floating-point arithmetic. We will make use of them by implementing a numerical method for integration.

Recall (or learn) that the integral $\int_a^b f(x)\,dx$ of a real-valued function $f(x)$ of one real-variable gives the signed area under the graph of f over the interval $[a, b]$. Most integrals do not have closed-form expressions as solutions and one must resort to numerical methods for their computation.

To this end, let us partition the interval $[a, b]$ into N subintervals $[x_0, x_1], \ldots, [x_{N-1}, x_N]$ with $x_i = a + i * \Delta x$, where $\Delta x = (b - a)/N$, and let x_i^* be the midpoint of the ith interval. Then the "midpoint method" approximates the integral according to the formula

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f(x_i^*)\Delta x \quad (2)$$

with the approximation becoming exact as $N \to \infty$.

Your task is to write a procedure that computes the definite integral of a real-valued function of one variable using the midpoint method. Your procedure should accept as input the address of the function to be integrated and the endpoints of the interval of integration (specified in the data section). The value of N should be hard-coded into the body of your method. You should have a helper procedure that checks that $a < b$, printing an appropriate error message if necessary. Since a, b and the result of integration will be floating point values, you will need to use the appropriate registers of the floating point coprocessor to manipulate them. Refer to the assignment template for additional details.

4 Assignment Submission Instructions

1. Submit your solution to MyCourses before the due date.
2. Highlight the Three template files, and zip them. The zipped file will be named <studentID>.zip. If my student ID is 123456789, then the zip file to submit will be named "123456789.zip".
3. If you have special comments about your code in any questions, feel free to include a "confessions.txt" file in your zip containing your specific comments. Otherwise, simply comment your code as you normally would.
4. Your code *must* run and assemble, even if the final solution is not quite working. Part marks will be awarded for correct high-level control flow and use of conventions. If something is not working, comment-out the broken parts of code and write a comment or two about what you expect to happen, what is happening, and what the problem may be. *If your code does not assemble you will receive very few points*