|  |  |  |
| --- | --- | --- |
| American University of Sharjah  College of Engineering  Department of Computer Science & Engineering  P. O. Box 26666, Sharjah, UAE |  | **Instructors:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-12  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Spring 2021 |

**CMP305L - Data Structures and Algorithms Lab**

**Lab. Assignment 9 – Application of Recursion**

***Objectives:***

To understand

* Program recursive functions with accumulators
* Implement recursive traversal of linked lists
* Develop programs that make use of recursion

***Note:***

***Lab:*** Exercises 1,2 and 3 (10 marks)

***Bonus*:** Exercise 4 (1 mark)

**Exercise 1**

Implement a recursive function *PI(n*) that calculates *π*with *n*terms, as per the series given below. Write two versions: (a) *without* accumulator and (b) *with accumulator.* Note that you need first to derive a proper recurrence formula, then implement it.

**π = 3** +

**Exercise 2**

Write a recursive function that mixes up the elements of two given linked lists i.e., taking an element alternately from one then from the other, etc. The lists may be of any length; the extra elements from the longer list should be appended at the end. Use singly linked lists from Lab 3.

*Examples:* head = alternateMix( headList1, headList2 );

If headList1 points to the list [3,4,7] and headList2 points to the list [2,5,6,8,] then head will point to a new list containing [3,2,4,5,7,6,8].

Likewise, mixing the two lists [2,4,6,8] and [3,5,7] will produce the new list [2,3,4,5,6,7,8], while mixing [7,1,4] and [2,8,9,5,3,6] will yield [7,2,1,8,4,9,5,3,6].

***Exercise 3:***

Using the given program demo/shell, write a recursive function that draws an *H-Tree* of given depth *N*. The five *H-Trees* for *N= 1* to *N= 5* are shown hereafter, for your reference.

Note that the task only requires to devise the recursive logic. All graphic code is included and you only need to use the *drawH()* and *drawline()* functions, as may be necessary.

|  |  |  |
| --- | --- | --- |
| http://www.itu.dk/people/kasper/BITP-2011/00_Revision%20exercises/recursive_procedures/applet/htree1.jpg | https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcSD5ZVuteLxDQ1F8hTNBOiiKH9BVndqTIycjpFBOg9DvRDS2ONm3g | http://www.cs.princeton.edu/courses/archive/fall02/cos126/assignments/htree3.jpg |
| http://www.cs.princeton.edu/courses/archive/spr01/cs126/assignments/htree4.jpg | http://www.cs.princeton.edu/courses/archive/fall03/cs126/assignments/htree5.png |  |

**Note:** This program uses the *simple-SVG* library from <https://github.com/adishavit/simple-svg>. Interestingly, the entire library is contained in a single header file: *simple\_svg\_1.0.0.hpp.*

Windows users: The library is included in the Visual Studio project given to you.

Mac users: You can use Xcode if you are familiar with it or, simply, you can compile the code in the Terminal with *g++ -std=c++11 -o demo demo.cpp* and run the program as *./demo.*

The easiest way to view the created SVG image is to drag and drop it in your web browser.

**Bonus:**

***Exercise 4:***

Write the below functions,

1. Write and test an accumulator recursive function that takes two vector iterators, it1 and it2 as parameters and returns the sum of the numbers of the vector between it1 and it2.

double calculate(vector<int>::iterator it1,

vector<int>::iterator it2, double sum=0)

1. Generalize the code in part (a) by making it a function templates, i.e.:

template< typename Iter, typename Value >

Value Sum( Iter it1, Iter it2, Value sum );

1. **Code:**

//Q4)a)

double calculate(vector<double>::iterator it1, vector<double>::iterator it2, double sum = 0) {

if (it1 >= --it2) return (sum);

sum += (\*it1)+(\*it2);

return calculate(++it1, it2, sum);

}

int main() {

vector<double> arr = { 1,2,3,4,5,6,7,8,9,10 };

cout << calculate(arr.begin(), arr.end()) << endl;

return 0;

}

**Screenshot:**



1. **Code:**

//Q4)b)

template< typename Iter, typename Value >

Value Sum(Iter it1, Iter it2, Value sum) {

if (it1 >= --it2) return (sum);

sum += (\*it1) + (\*it2);

return calculate(++it1, it2, sum);

}

int main() {

vector<double> arr = { 1,2,3,4,5,6,7,8,9,10 };

cout << Sum(arr.begin(), arr.end(),0.0) << endl;

return 0;

}

**Screenshot:**

