**Computer Programming 2**

**Lab 4**

**This assignment tests the concepts of**

* functions
* arrays
* vectors
* dynamic memory

**Objective**

We will be implementing a few functions again. The provided main.cpp file will be used to TEST your functions.

It is recommended that for each function you get the simplest case working and add handling of the other cases once everything else works. This is called stepwise refinement.

Implement each function prototyped in **pointers.h**. I recommend you start with the starter code for **pointers.cpp**. Read the provided preconditions and postconditions, ask questions if you do not understand them. Your implementation will be checked for compliance to these conditions. You may assume the precondition is ALWAYS met, you do not need to confirm the precondition is true.

You may modify the main function, however, you will only hand in functions.cpp. You will want to avoid modifying functions.h, since modifications here may make your code unable to be compiled by the grader.

**starter pointers.cpp**

#include <iostream>

#include "pointers.h"

int \*sub\_array(int \*start, int \*end) {

**return** **nullptr**; *//***TODO***: stub*

}

unsigned int len(int \*start, int \*end) {

**return** 0; *//***TODO***: stub*

}

void printNonNegArray(const int\* array) {

}

void printVector(const std::vector<int>& array) {

}

void reverseVector(const std::vector<int> &source, std::vector<int> &destination) {

}

**Example Output**

test1: [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

test2: [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

length: 5

test4: [4, 8, 16, 32]

test5: [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2]

test6: [2]

**Deliverables**

* Implementation of Functions: **pointers.cpp**

**Submitted assignments without the above file named correctly will render your assignment as uncompilable and will be detrimental to your assignment grade.**

main.cpp

#include <iostream>

#include <vector>

#include "pointers.h"

using namespace std;

int main() {

vector<int> va({2, 4, 8, 16, 32, 64, 128, 256, 512, 1024});

vector<int> va2({2});

//The negative 1 marks the end of the array

int ia[] = {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, -1};

cout << "test1: ";

printNonNegArray(ia);

cout << endl;

cout << "test2: ";

printVector(va);

cout << endl;

cout << "length: " << len(ia, ia+4) << endl;

int \*sub = sub\_array(ia+1, ia+4);

cout << "test4: ";

printNonNegArray(sub);

cout << endl;

//Reclaim storage by the dynamic array

delete [] sub;

vector<int> var;

reverseVector(va, var);

cout << "test5: ";

printVector(var);

cout << endl;

reverseVector(va2, var);

cout << "test6: ";

printVector(var);

cout << endl;

return 0;

}

pointers.h

#ifndef LAB\_POINTERS\_H

#define LAB\_POINTERS\_H

#include <vector>

//Precondition: none

//Postcondition: destination has (2n-1) items of source. The elements from position 1 to end of source

// are duplicates at the end of the vector in reverse order. The last element of source is NOT duplicated.

// [8, 2, 5] => [8, 2, 5, 2, 8]

void reverseVector(const std::vector<int>& source, std::vector<int>& destination);

//Precondion: array points to a integer array where the last element is a -1

//Postcondition: Outputs the array in Python formatting, such as [8, 2, 5, 2, 8]

void printNonNegArray(const int\* array);

//Precondion: n/a

//Postcondition: Outputs the array in Python formatting, such as [8, 2, 5, 2, 8]

void printVector(const std::vector<int>& array);

//Precondion: start and end point to elements in an array. end > start. end is inclusive.

//Postcondition: Returns the number of elements in the array

unsigned int len(int\* start, int\* end);

//Precondion: start and end point to elements in an array. end > start. end is inclusive.

//Postcondition: Creates a dynamic array on the heap of the exact size required to hold start to end.

// Returns the address to the new dynamic array.

int\* sub\_array(int\* start, int\* end);

#endif //LAB\_POINTERS\_H