BTN415 Lab 7 – Pointers and Memory

In this lab, you will work with a linked list containing elements with dynamic memory allocation.

LEARNING OUTCOMES

Upon successful completion of this lab, you will have demonstrated the ability to:

- Work with pointers
- Work with dynamic memory allocation
- Work with Linked Lists

For this lab, you should create methods in a file called vectors.cpp (partially written in https://github.com/marceljar/BTN415_Labs/blob/main/lab7/vectors.cpp). This file will define methods declared in *vectors.h* (https://github.com/marceljar/BTN415_Labs/blob/main/lab7/vectors.h) . Then, you can test your results by running them with the main source code file, which is provided in https://github.com/marceljar/BTN415_Labs/blob/main/lab7/main.cpp. In the provided *vectors.h* file, two classes are defined: Vector and List, whose definitions are shown below:

```
class Vector {
private:
	std::string name;
	Vector* next;
	int size;
	int* elements;
public:
	Vector();
	Vector(std::string);
	~Vector();
	friend void List::add_vector(std::string);
	friend bool List::remove_vector(std::string);
	friend bool List::append_vector(std::string);
	friend bool List::print_vector(std::string);
	friend void List::print_vectors();
};
```

and

```
class List {
private:
	Vector* head;
	Vector* current;
public:
	List();
	void add_vector(std::string);
	bool remove_vector(std::string);
	bool append_vector(std::string);
	bool print_vector(std::string);
};
```

A description of each method that needs to be defined, as well as the number of marks assigned to each one, is provided in what follows.

PART A – [1.0 marks]

void add_vector(std::string)

This method should take as argument a string and create a new **Vector** node at the end of the linked list, (or create the first **Vector** node in case the list does not exist, with a **name** property equal to the provided argument. Note that a constructor for **Vector** nodes is already provided! Your task is to call it properly and make sure that the list is properly linked. This method returns *void*.

PART B – [1.5 marks]

bool remove_vector(std::string)

This method should take as argument a string and then removes the **Vector** node that has a name that matches the provided argument. This method returns *true* if a **Vector** node was found with a name that matches the string, and *false* otherwise. Note that a destructor for Vector nodes is already provided! Also, care must be taken to keep the list properly linked.

PART C – [1.5 marks]

bool append_vector(std::string)

This method should take as argument a string, and then ask the user for the number of integers to add at the end of the Vector node. Then, your method should make sure to:

- Change the **size** property of the vector
- Reallocate memory in order to create a new set of vector **elements** containing all previous integers as well as all new integers in your vector
- Get rid of any no longer used memory. I.e., avoid memory leakage.

This method returns *true* if a **Vector** node was found with a **name** that matches the string, and *false* otherwise.

PART B – [1.0 marks]

bool print_vector(std::string)

This method should print all **elements** of the vector which **name** property matches the provided argument. This method returns *true* if a **Vector** node was found with a **name** that matches the string, and *false* otherwise.

SUBMISSION INSTRUCTIONS

You only need to submit one source code: a definitions file vectors.h.