BTN415 Lab 8 – Serializers and De-serializers

In this lab, you will create functions that will allows packets with dynamic contents to be transmitted over sockets.

# 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 serializers and de-serializers

For this lab, you should create serializer and de-serializer methods in a file called socket*.*cpp (which was covered in class, see <https://github.com/marceljar/BTN415_Labs/blob/main/lab8/socket.cpp>). This file will define serializer and de-serializer methods that should be declared in a socket*.h* header file(also covered in class, see <https://github.com/marceljar/BTN415_Labs/blob/main/lab8/socket.h>), to transmit packets with a structure that is given below:

struct data\_packet {

char name[16];

int id;

int size;

int\* numbers;

int tail;

};

You can test your code using the *client.cpp* and *server.cpp* files provided in the links below (make the proper changes from **packet** to **data\_packet** for testing purposes):

<https://github.com/marceljar/BTN415_Labs/blob/main/lab8/client.cpp>

<https://github.com/marceljar/BTN415_Labs/blob/main/lab8/server.cpp>

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

## PART A – [0.5 marks]

### data\_packet create\_data\_packet()

This function (which should not belong to any class, but rather be a regular function), should ask the user to enter all required information to create a new **data\_ packet**. Note that the tail could be any integer that you want. *Hint: There is no need to sanitize user input. You can assume that the user is always entering valid data.*

Text

Description automatically generated

## PART B – [2.0 marks]

### serialized\_packet data\_packet\_serializer(data\_packet)

This function (which should not belong to any class, but rather be a regular function), should take as an argument a **data\_packet**. Then, as the name suggests, it should save all contents of the **data\_packet** (including dynamically allocated ones) into a *char array*. Finally, the function should return a **serialized\_packet** struct containing the char array as well as the length of this char array (i.e., the total size of the packet in bytes).

Text

Description automatically generated

## PART C – [1.5 marks]

### data\_packet data\_packet\_deserializer(char \*)

This function (which should not belong to any class, but rather be a regular function), should take as an argument a *char array*. As the name suggests, this function should save all contents of the *char array* in the proper fields inside a **data\_packet** struct which, in turn, it then returns as an output.

Text

Description automatically generated

## PART D – [1.0 marks]

### int send\_data\_packet (data\_packet) (for both the Server\_TCP as well as Client\_TCP classes)

This method, as its name suggest, should send a **data\_packet** over a socket socket and return the number of transmitted bytes. *Hint: Use the data\_packet\_serializer() function to accomplish your goals.*

*A screenshot of a computer

Description automatically generated with medium confidence*

*Text

Description automatically generated*

### int receive\_data\_packet (data\_packet&) (for both the Server\_TCP as well as Client\_TCP classes)

This method, as its name suggest, should receive data sent over a socket, and save it in the proper fields of a **data\_packet** struct, as well as return the number of transmitted bytes. *Hint: Use the data\_packet\_deserializer() function to accomplish your goals.*

*Text

Description automatically generated*

*Text

Description automatically generated*

**Output: Server + Client code**

*A screenshot of a computer

Description automatically generated with medium confidence*

### SUBMISSION INSTRUCTIONS

*You only need to submit the updated socket.cpp and socket.h files.*