Submit 2 files “helper.c” and “header.h” in a folder “impl”. The “impl” must be in a folder named BITSID_CNLab7(2021A7PS1234G_CNLab7). Submit a zip of the outer folder. Do not use the code given in GitHub. There should be no other files. You can create a main.c for your testing, **do not submit it**. The helper.c must contain the implementation of the following functions. Main function is not necessary. You can add your own extra functions but nothing has to be printed to stdout. Since you should already be familiar with the network system calls, this lab won’t be testing those.

What to implement:

1. `unsigned char* serialize( struct packet* p):` Function for serializing the packet using the same header structure as mentioned in original document **(1 mark)**

2. `struct packet* deserialize(unsigned char* buffer):` Function for deserializing the packet **(1 mark)**

3. `int checksumCalc(unsigned char* buffer):` After serializing all the other fields, call this function from inside the serialize function by passing the buffer to calculate the checkSum. Hint: The 2nd index(starting from 0 index) of the buffer must be right-shifted by 2 during checksum calculation. **(1 mark)**

4. `unsigned char* getPacket(char* input, int srcCampus, int srcDept, int* validDept[3], int numOfValidDept[3]):`Identify the type of packet that has to be sent based on the input and return the required serialized byte stream by calling `generateUnicastPacket(...)` if it’s a unicast packet, `generateBroadcastPacket(...)` if it’s a broadcast packet and `generateControlPacket(...)` if it’s a control packet. For testing purposes, the function will be called directly with the testing input(You don’t have to take user input from the terminal). **(0.5 marks)**

5. `unsigned char* generateUnicastPacket(char* input, int srcCampus, int srcDept, int* validDept[3], int numOfValidDept[3]):` Based on the input which will be of the form "1.<dest_campus>:<dest_department>:<payload>\n" or “2.<dest_department>:<payload>\n” **(1 mark)**
   Here, dest campus and dest department will be numbers and not strings. Tokenize the input, generate the packets, serialize it and return the byte stream. The first element in validDestDept contains an int pointer which points to the starting address of an array containing all valid departments connected to the Pilani server, the second to the Goa server and the third to the Hyderabad server. The lengths of each of the above mentioned arrays are given in the second array. If any src/dest campus/dept mentioned as part of the arguments is not valid, return a null pointer. Consider hops, ACK and type fields as “don’t care” values for the take home assignment. Although they maybe “‘don’t

care” values, the serialize/deserialize functions should still properly encode/decode those fields.

6. `unsigned char* generateBroadcastPacket(char* input, int srcCampus, int srcDept, int* validDept[3], int numOfValidDept[3]):` Based on the input which will be of the form "3.<payload>\n" or "4.<payload>\n" . Same explanation as above. **(1 mark)**

7. `unsigned char* generateControlPacket(char* input, int srcCampus, int srcDept, int* validDept[3], int numOfValidDept[3]):` Based on the input which will be of the form "5.EXIT\n". Same explanation as above. The type field should be 1 for EXIT. **NOTE: LIST does not have to be implemented. (0.5 mark)**

Example of input:

`generateUnicastPacket(...)` could be called with input as “1.2:1:Hello\n” , srcCampus as 0, srcDept as 5, validDestDept as [[1,3,5], [2,4], [1,2,3]], numOfValidDestept as [3,2,3]. Now the input implies destDept 1 of destCampus Hyd. Since srcDept 5 is a valid department in campus Pilani, the packet should be generated populating all the fields including the campus fields(since message is of type inter-campus) and the payload and serialized and the byte stream be returned. Had atleast one input been invalid, a null pointer should have been returned.