MSSV: 24127230

Tên: Nguyễn Kinh Quốc

1. The size in bytes of that data structure is **120.** Because of the struct alignment, the first array contains 100 char, so the size of it is 100 (which is stored from the first cell to the 99th cell). Thanks to that, the next data type, which is int (its size is 4 bytes), is going to be stored at the address 100th to 103th, which is possible because 100 is divisible by 4. Next, the float data type is stored from 104th to 107th within the same reason. Finally, with the pointer class, its size is 8 so it has to be stored at the address which is divisible by 8 and also is closest to the 107th, and it is 112th, and then we add up 8 to 120. So the final result is 120 bytes.

2. The first variables (which is a) would occupy more space because it would occupy 120 bytes in total, comparing to *b which only occupies 8 bytes as it only contains the address of the struct.

3. All of them would occupy 8 bytes. As it only contains the address.

4. The name would be stored from 100 - 199, then the staff_id and height would respectively be stored from 200 - 203 and 204 - 207 and the department_information would be stored from 208 - 215. Using the struct alignment logic in the first question.

5. It would be from 100th to 107th. As the size of it is 8 bytes.

6. Retaining the first line and then delete one asterisk on the second line. The first does match the requirement, whilst the second one actually is the pointer to pointer of struct c_ptr.

7. Yes, the following assignments can be made successfully. On the first line, c is initialized as a struct type, and then the c_ptr on the second line would point to the address of c. Then, it initialized d

and the pointer d_ptr would point to c_ptr address which would point to c. So all of them are valid, but perhaps it will not match the intention of the coder.

8. It is incorrect. If it were correct, it would point to the address of c struct. As c is a struct type, not a pointer, so dereferencing c would be invalid, instead we should change the asterisk to the reference

9. It would return the address of the c.dept_ptr. Since the c.dept_ptr has not been initialized yet. So it would return the address of a random place.