1. a) In file contact.h (line 15)

incorrect => char middleInitial[5];

correct => char middleInitial[7];

The size of array given was 5 and we are scanning

scanf("%6[^\n]%\*c", name->middleInitial); 6 characters and it also need /0(null terminator) at the end for string. So, we need 1 more size for it. That why, size will be 7 for middleInitial not 5.

b) In file contact.c (line 31)

incorrect => printf("Do you want to enter a middle initial(s)? (y or n): "); scanf("%s%\*c", &yesNo);

correct => printf("Do you want to enter a middle initial(s)? (y or n): "); scanf("%c%\*c", &yesNo);

We don’t need to scan a string. Just a single character (y or n). and even we are using char yesNo which is fixed 1 character not an array. That why, the type specifier will be %c not %s, which takes single character.

c) In file contact.c (line 57)

incorrect => if (yesNo == “y” || yesNo == “Y”)

correct => if (yesNo == 'y' || yesNo == 'Y')

“” double quotes will take for string literals with null terminator. We don’t want to compare string. Just a character. So, we use single quotes to compare character.

d) In file contact.c (line 86)

incorrect => if (yesNo == 'y' && yesNo == 'Y')

correct => if (yesNo == 'y' || yesNo == 'Y')

we need to use OR operator. To satisfy the condition. &&(and) will require both to be true.

And we need any one to be true. Therefore, OR is used.

1. In memory things go as declared. So, first Contact struct will be laid. Contract struct consists of other structs as member. So, in sequence, first struct name, second struct address and then struct numbers. And when any this struct goes it goes with its own members in memory. If we want to use firstname member of Name struct then we do by using dot to access it. And similarly, we do with other inside members.
2. I will create a function which will take address of struct name, address, and number. Because this is being repeatedly.

Prototype ==>

Void getcontact(struct Name\*, struct Address\*, struct Number\*);

So whenever getcontact is called it passes it address and receives the address in pointer form. I have used void which means to value will be returned.

4. scanf takes every character even which we press enter or its new line if we use “%c” only. So, to stop taking those character and to stop going into buffer, we use %\*c which tells that to not to take newline in variable storage. So, this will stop taking characters after we press enter and also not take enter as its character.

5. %[^/n] will scan until it gets new line character. Which means until new line starts. And it can take whitespace also. But if we think of %s it discards whitespace. In current situation we are scanning

(N. O.) which is having whitespace. So %s will stop after (N.). That’s the reason why we use %[^/n] not %s.