Homework – reference in cpp

Answers:

1. a reference in ccp is basically a duplicate of a chosen variable, it is also still linked to the variable so when it gets updated or modified so does the variable and vice versa.

2. references are more readable than most other methods of passing parameters to functions and they are also generally considered more efficient seeing as using them is like using the original variable itself while other method usually involve taking some steps until reaching the required information.

3. one advantage of using references over pointers is that it doesn't require any copying of variables when passing a parameter to a function, with pointers when you run the risk of copying code when dereferencing the pointer whereas when using references once you initialize the reference there is no more copying. Another advantage of references is that they are generally considered safer. references can't point to null, can't be reassigned and have to be initialized which are all thing pointers can do and this can lead to problems while using pointers if you're not careful.

4. out of the three presented ways to call using reference the only correct one is square(3, y);. The option square(3, &y); doesn't work because we don't need the address of y when using reference and the option square(3, 6); doesn't work because in the function we try to give a value to result and if we pass six which is an int and can't be modified then we would get an error.

5. in the first bit of code the problem is that you're trying to return an int that was created in the function by reference but because the int was created in the function once the function finishes running the int will get deallocated leaving the reference copying empty space and this will probably lead to errors.

In the second function the problem I found is that because the delete command needs a pointer to delete allocated memory, once you return a reference to the allocated memory the pointer is lost and there is no way to delete the allocated memory causing memory leakage.