CS 111

reference parameters

Call by value

ANSWER_TYPE FUNCTION_NAME(LIST_OF_PARAMETER_DECLARATIONS)

- Each parameter declaration specifies a parameter name and type, for example int x
- Normally, when the function is called, a parameter is created and its initial value is a copy of the value of the argument it is used for
  - This is known as *call by value*

Call by reference

- If we introduce an & between the type and name of the parameter it applies *call by reference*
- This means that the argument must be a variable and that variable is used inside the function whenever we make reference to the parameter
- This means that changes to the parameter are permanent changes to the calling argument

Key summary

- For a *call by value* parameter:
  - A copy of the value is passed
  - Changes made to the value inside the function are not permanent
  - A calling argument can be a hard-coded number, for example
    - e.g., sqrt(5.0);
- For a *call by reference* parameter:
  - Changes are permanent
  - A calling argument must be a variable

Example 1

```
void makeChanges(int &x, int y){
    y = x + y;
    x = y;
}

int main(){
    int a = 5, b = 6;
    makeChanges(a, b);
    cout << a << " " << b;
}
```

Example 2

```
void makeChanges(int &x, int y){
    y = x + y;
    x = y;
}

int main(){
    int a = 5, b = 6;
    makeChanges(a, b);
    makeChanges(a, b);
    cout << a << " " << b;
}
```

Example 3

- Predict the output at line a
- What is the bug in the following main function?

```
void makeChanges(int &x, int y){
    y = x + y;
    x = y;
}
int main(){
    int x = 6, y = 7;
    makeChanges(x + 1, y + 1);
    cout << x + 1 << " " << y + 1 << endl; // line a
    return 0;
}
```