Modern C++ Overview
Part Two Solutions

Lambda Expression

- Briefly describe what is meant by a lambda expression
  - A lambda expression is an anonymous, inline function. It is used to create a local function, mainly for passing as an argument to a function call or as a return value

Defining a lambda expression

- Briefly describe the syntax for writing a lambda expression
  - We put [] for the function name
  - The arguments are written in the usual way
  - The body is written in the usual way, as an inline function
  - The compiler will deduce the return type (except in C++11, if the function body returns a value and contains more than one statement)
- Write down a lambda expression that takes an int argument and returns double the value of the argument

```
[] (int arg) { return 2 * arg; }
```

Example of lambda expression usage

- The C++ standard algorithm function count_if takes three arguments: the begin and end of an iterator range, and a predicate function which returns a boolean
- It calls the predicate function on every element in the iterator range
- Use count_if() to write a program which prints out the number of odd elements in a vector of int, using a suitable lambda expression

Capture

- Briefly explain what is meant by "capture" in a lambda expression and how to implement it
  - A capture makes variables in the local scope available for use in the body of the lambda expression
  - This is done by writing the names of the desired variables inside the [] of the lambda expression
  - By default, variables are captured by value
  - To capture a variable by reference, put a '&' in front of its name

Capture

- Write down lambda expressions which capture a local variable x
  - By value

    ```
    [x]() { /* Use copy of x */ }
    ```

  - By reference

    ```
    [&x]() { /* Use reference to x */ }
    ```

Capture all local variables

- Write down lambda expressions which capture all local variables
  - By value

    ```
    [=]() { /* Use copies of local variables */ }
    ```
  - By reference

    ```
    [&]() { /* Use references to local variables */ }
    ```

Capture and objects

- Write down lambda expressions which could be used in a member function to capture the data members of the object

  [this]() { /* Use references to data members */ }

  [this]() { /* Use references to date variables */ }

- How does this differ from capturing local variables?
  - The data members are captured through a reference to the object (by dereferencing the "this" pointer)
  - No special syntax is needed to modify the data members

Example of lambda expression with capture

- Alter the earlier count_if example so that it finds the number of exact multiples of any integer (instead of the hard-coded value 2)
- The integer will be a local variable which is captured by the lambda expression
- Write a program that uses this lambda expression to find the number of exact multiples of 3