Practical 3 Questions

1. What are the two principal characteristics of a recursive algorithm?
   A.The algorithm must have a base case which is reachable.
   B.Have a recursive step which will call upon itself given that the base case isn't true.

2. Recursion is..
   Answer D:
   Theoretically powerful and often used in algorithms that could benefit from recursive methods

3. True:
   All recursive functions can be implemented iteratively.
   A special case of recursion: tail recursion. This is where the recursive call is the last thing executed by the function. This type of recursion can be easily translated into a loop.

4. False:
   Compiler will not detect this however during execution a runtime error will be thrown as the algorithm will exceed the size of the stack and cause a stack overflow.

5. False not all recursive functions return void:
   An example being the factorial recursive function where we return
   n = n*(factorial(n-1)); - here the return type is int.

6. False Recursive calls are not always contained in a loop.

7. True: The function will constantly call itself as it has no base case.

8.

```
int mystery(int n)
{
        if (n>0) return n + mystery(n-1);
        return 0;
}
```

B:The base case for this recursive function is an argument with the value zero.

9. **Bugs associated with recursion:**
   Memory issues- no base case
   Base case is checked after recursive step -> can never be reached
   Recomputation for things such as fibonacci recursive implementation
   Sub problem doesn't grow closer to the base case

10. Memoization: Where the values of expensive function calls are stored in something like an array, and when the same function call is made instead of computing it returns to the array and gets the value instead.