COMP2041 Week 7 Tutorial

- talk about my - global.pl
- talk about add, cmp
- be consistent whether to use brackets or not, though I do admit that I omit it if I can
- convention about names (just be super consistent camel case or snake case)

Question 2

```
%count_words = ();
@lines = <STDIN>;
$line_str = lc join '', @lines;

foreach $word (split /\W+/, $line_str) {
    $count_words{$word} += 1;
}

# it's sort of like anonymous function
foreach $key (sort {$count_words{$a} <=> $count_words{$b}} (keys %count_words)) {
    print "$count_words{$key} $key\n";
}

# using subroutine
sub by_count {
    $count_words{$a} <=> $count_words{$b};
}

foreach $key (sort by_count (keys %count_words)) {
    print "$count_words{$key} $key\n";
}
```

- can supply a subroutine as the first parameter to sort

Question 3

```
%words = ();

open FILE, "<$ARGV[0]" or die;
while (<FILE>) {
    chomp;
    foreach $word (split /\W+/) {
        $words{$word} = 1;
    }
}
```

```
close FILE;

open FILE, "<$ARGV[1]" or die;
while (<FILE>) {
    chomp;
    foreach $word (split /\W+/) {
        delete $words{$word} if exists $words{$word};
    }
}
close FILE;

foreach $word (sort keys %words) {
    print "$word\n";
}
```

- difference between exists and defined (in the picture)

Question 5

```
sub printHash {
    my (%hash) = @_;
    my $n = 0;
    while (($key, $value) = each %hash) {
        printf "[%s] => %s\n", $key, $value;
        $n++;
    }
    return $n;
}

%colours = ("John" => "blue", "Anne" => "red", "Andrew" => "green");
$n = printHash(%colours);
print "There are $n items in our colours\n";
```

Question 6

```
%bigrams = ();
$prev_word = "";

while (<STDIN>) {
    chomp;
    foreach $word (split /\W+/) {
        $word = lc $word;
        $bigrams{$prev_word}{$word}++ if $prev_word;
        $prev_word = $word;
    }
}
```

```
foreach $prev_word (sort keys %bigrams) {
    $total = 0;
    $max_word = "";
    $max_count = 0;

    for $word (keys %{$bigrams{$prev_word}}) {
        $total += $bigrams{$prev_word}{$word};
        if ($max_count < $bigrams{$prev_word}{$word}) {
            $max_count = $bigrams{$prev_word}{$word};
            $max_word = $word;
        }
    }
    printf "%s(%d) %s(%d)\n", $prev_word, $total, $max_word, $max_count;
}
```