CS2205: Theory of Computation
Problem Class XIV

1. Suppose B is a Boolean expression that does not contain any procedure calls. Hence, give a big-Oh characterisation, in terms of n, of the worst-case time complexity of the following:

```
if B
{
}
```

2. Now suppose that B *does* contain a procedure call, and this procedure has worst-case time complexity $O(h(n))$. Hence, give a big-Oh characterisation, in terms of n, of the worst-case time complexity of the algorithm in the previous question.

3. Suppose B is a Boolean expression that does not contain any procedure calls. Hence, give a big-Oh characterisation, in terms of n, of the worst-case time complexity of the following:

```
if B
{    ...something that is O(f(n));
}
else
{    ...something that is O(g(n));
}
```

4. Now suppose that B *does* contain a procedure call, and this procedure has worst-case time complexity $O(h(n))$. Hence, give a big-Oh characterisation, in terms of n, of the worst-case time complexity of the algorithm in the previous question.

5. Give a big-Oh characterisation, in terms of n, of the worst-case time complexity of a **while** loop whose loop test is known to be always **false**, e.g.:


```
while 1 ≠ 1
{    ...something that is O(f(n));
}
```

6. Dr. Toc is told about an algorithm whose worst-case time complexity is $O(1)$. "That means," the good doctor says, "that the algorithm consists of just one single command."

   Is the good doctor right or has the good doctor been hitting the bottle again? Explain your answer in detail.

7. For each of the following algorithms, give a big-Oh characterisation, in terms of n, of the worst-case time complexity of the algorithm. In each case, justify your answers.

```
Algorithm:  A(n)
s := 0;
for i := 1 upto n
{    s := s + 1;
}
```

```
Algorithm:  B(n)
s := 1;
for i := 1 upto 2n
{    s := s × i;
}
```

```
Algorithm:  C(n)
s := 0;
for i := 1 upto n
{    for j := 1 upto n
     {    s := s + i × j
     }
}
```

Algorithm: D(n)

```
s := 0;
for i := 1 upto 2n
{    for j := 1 upto n
     {    s := s + i × j
     }
}
```

Algorithm: E(n)

```
s := 0;
for i := 1 upto n
{    for j := 1 upto i
     {    s := s + i × j
     }
}
```

Algorithm: F(n)

```
s := 0;
for i := 1 upto 2n
{    for j := 1 upto i
     {    s := s + i × j
     }
}
```

Algorithm: G(n)

```
s := 0;
for i := 1 upto n
{    for j := 1 upto i^2
     {    s := s + i × j;
     }
}
```


Algorithm: H(n)

```
s := 0;
for i := 1 upto 100
{    for j := 1 upto i
     {    for k := 1 upto n
          {    s := s + i × j + k;
          }
     }
     for m := 1 upto n
     {    s := s + s;
     }
}
```