It is dangerous to compare a signed integer to an unsigned integer because:
a) signed integers can only represent half as many values as unsigned.
b) unsigned integers cannot represent fractions to enough precision.
c) there are no negative unsigned integers.
d) type casting slows computation.

What does this program print?

```
int main()
{
    char p;
    char buf[10] = {1, 2, 3, 4, 5, 6, 9, 8};
    p = (buf + 1)[5];
    cout << (int) p << endl;
    return 0;
}
```

a) 5
b) 6
c) 9
d) Error
e) None of the above

What will be printed:

```
int main()
{
    char str1[] = "abcd";
    char str2[] = "abcd";
    if (str1 == str2)
        cout << "Equal" << endl;
    else
        cout << "Unequal" << endl;
    return 0;
}
```

a) Equal
b) Unequal
c) Error
d) None of these.

An array passed as an argument to a function is interpreted as
a) the address of the array.
b) the value of the first elements of the array.
c) the address of the first element of the array.
d) the number of elements in the array.

What is the output of the program:

```
int main()
{
    int a = 11, b = 5;
    if (a = 5)
        b++;
    cout << ++a << " " << b++ << endl;
    return 0;
}
```

a) 12 7
b) 5 6
c) 6 6
d) 6 7
e) 11 6

What will happen after attempting to compile and run the following code?

```
int main()
{
    cout << main << endl;
    return 0;
}
```

a) Will not get past the compiling step due to an error.
b) Will result in an infinite loop.
c) "main" will be printed.
d) The address of main will be printed.
e) A crash will occur.
f) None of these.

"Good style" says you should:
a) have as many returns from a function as you want
b) have as few returns from a function as practical, hopefully just one
c) never have more than one return from a function
d) (a) and (b)
e) (b) and (c)
f) (a) and (c)