Prepare your answers to the following questions in a plain text file. Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted. For all questions, show supporting work if you want credit.

You will submit your answers to the Curator System (www.cs.vt.edu/curator) under the heading `HW02`.

For questions 1 through 5, translate each of the given C statements into valid MIPS assembly language code. Some statements can be translated into a single MIPS instruction; some will require a sequence of two or more MIPS instructions. In each case, take the variable names used in the C statement as indicators of which MIPS registers you should use.

You must not violate the semantic integrity of the given C statements. For example, if a C programmer writes the following code

```
int x = 5, y = 10, z = 20;

z = x++ + y;
```

then that programmer would expect that `z` would equal `15`, `x` would equal `6` and `y` would equal `10` because those results are consistent with the code and with the definition of the C language. Be careful of this… there is one real subtlety in one question.

You should not modify registers, or memory contents, that do not correspond to C variables that would be modified. For example, your solution to question 1 should not modify `$s0`, `$s4` or `$s5`, or any other register than `$s1`. However, your solutions are free to modify the register `$at` as needed.

You may assume the C variables have been declared as follows, and that they've all been initialized before being used:

```
int  s0, s1, s2, s3, s4;
int* s5;
```

1. [20 points]
```
s1 = 8 * s2 + s2 * s3;
```

2. [20 points]
```
s0 = *s5;
```

3. [20 points]
```
*s5 = s1;
```

4. [20 points]
```
if ( s0 > s1 ) {
    s2 = ++s3;
}
else {
    s3 = s2++;
}
```

5. [20 points]
```
while ( s0 <= s1 ) {
    s3 = s2 + *s5;
    --s1;
}
```