Part I:

1.

Ls -l gets:

```
 -rwxr-xr-x 1 hunter2e temp 16696 Nov 8 11:10 1.out
```

Size 1.out gets:

```
 text    data     bss     dec     hex filename
1569     600       8    2177     881 1.out
```

2.

Ls -l gets:

```
-rwxr-xr-x 1 hunter2e temp 16720 Nov 8 11:19 2.out
```

Size 2.out gets:

```
   text    data     bss     dec     hex filename
   1569     600    4032    6201    1839 2.out
```

3.

Ls -l gets:

```
-rwxr-xr-x 1 hunter2e temp 20736 Nov 8 11:23 3.out
```

Size 3.out gets:

```
   text    data     bss     dec     hex filename
   1569    4616       8    6193    1831 3.out
```

4.

Ls -l gets:

```
-rwxr-xr-x 1 hunter2e temp 20816 Nov 8 11:31 4.out
```

Size 4.out gets:

```
     text    data     bss     dec     hex filename
```

```
   1814    4624       8    6446    192e 4.out
```

It seems data stored locally in a function is not stored in the executable in the same way as when it is declared globally. Whether or not it is initialized seems to have no major effect on the size of the executable.

5.

```
gcc -O3 -o 5o.out 5.c
```

AND

```
gcc -g -o 5d.out 5.c
```

Ls -l gets:

```
-rwxr-xr-x 1 hunter2e temp 23480 Nov 8 11:42 5d.out
```

Size 5d.out gets:

```
   text    data     bss     dec     hex filename
   1814    4624       8    6446    192e 5d.out
```

Ls -l gets:

```
-rwxr-xr-x 1 hunter2e temp 20776 Nov 8 11:44 5o.out
```

Size 5o.out gets:

```
   text    data     bss     dec     hex filename
   1659    4616       8    6283    188b 5o.out
```

The a.out file size is affected by compiling for debugging, but the segments are not. As for optimization the text seems to be the most affected as well as minimally lower values across the field.

Part II:

Printed after running:

The global variable (found to be stored in data) address is 0x563ad37e8010

The stack top is near 0x7ffe06d3aa20

Adding to bss with variable j (found to be stored in bss segment): 0x7ffe06d3aa24h

Part III:

1.

Line
Start 9 (main)
Call at: 10
Local variable stored: i (int(1))
Argument makes call jumps back to 1
Local variable stored: i (int(0))
Not greater, printf called line 5
Returns to line 3
Function complete returns to line 10
Program Complete

2.

Functions used to review stack:

```
gcc -g main.c
gbd a.out
break 11 (stop before end of program)
Info frame (shows stack)
```