1. \( x = ++a[1]; \) For this one, \( a[1] \) is incremented \textit{before} getting assigned to \( x \). That is, what happens here is \( a[1]++ \) followed by \( x = a[1] \). So, \( x = 2 \), and \( a[1] \) becomes 2.

\( y = a[1]++; \) Here, what happens is: \( y = a[1] \) and then \( a[1]++ \). So, \( y = 2 \) and \( a[1] \) becomes 3;

\( z = a[x++] \); This is equivalent to \( z = a[x] \) followed by \( x++ \). This means \( z = z[2] = 15 \) and \( x \) becomes 3.

\textbf{The output is:} 3 2 15

2. Doing it by hand: It is faster and you have full control.
   Garbage collector: removes the burden from the programmer.

3. \texttt{unsigned compare(int x, int y){}
   \hspace{1em} return !(x^y);}

4. a) \( y = 7*x; \)
   \hspace{1em} \( y = (x<<3) - x; \)

b) \( y = 27*x; \)
   \hspace{1em} \( y = (x<<5) - (x<<2) - x; \)

c) \( y = 67*x; \)
   \hspace{1em} \( y = (x<<6) + (x<<1) + x; \)
5. What are the values of M and N: **M = 15 and N = 9**

Show how did you reach your answer:
- An int is 4 bytes
- See the explanation next to the assembly code above
- Accessing array1[i][j] = array1 + (#elements in a row * 4*i) + (j*4) = array1+4Ni+4j → array2+36i+4j → N = 9
- Same array2[j][i] = array2 +4Mj+4j → array2+60j+4i → M = 15

6. From the assembly code and since both fun1 and fun2 have the same arguments in the same order:
- edx → a
- eax → b
- The result of the function is returned in eax
- cmpl %eax, %edx → edx – eax → a – b
- Combining the above bullet with jg2 .L9 we find that it does:
  if( a >= b) return b → if (a < b) return b → so fun1