```
Lauren Fromm
404751250

Assignment 1 - keys

1.1
1.
2.C-s P U B L I C enter
3.C-s L a enter M-b
4. C-s s e l f enter M-b
5. C-s a r r o w enter M-b
6. C-e
7. C-a
8. Yes, C-s searchs for the word.
   You can move forward or backward by using M-b and M-f.
   You can move to the front of the line using C-a and to the end
using C-e.
9. No, I used the commands above
9. C-z

1.2
2. C-s < ! - - C-a C-k
3. C-s < ! - C-a C-k
4. C-s < ! - C-a C-k M-f C-a C-k C-n C-a C-k
5. C-s < ! C-b C-b C-M-k C-d
   C-s < !  C-a C-k

1.3
2.C-s A M-f C-f C-d 3 7 C-s A M-f C-f C-d 3 7
3. C-s U T M-d M-d U S - A S C I I
4. C-s < / o M-a Enter
5. C-x C-x y

1.4
3. C-x ] C-s < ! - C-s C-s ent C-a C-d C-a C-SPC C-e M-w
C-s L a b ent C-e ent C-n C-y
C-s < ! - C-s C-s ent C-a C-d C-a C-SPC C-e M-w
C-s H o m C-s C-e enter C-n C-y
C-s < ! - C-s C-s C-s ent C-a C-d C-spc C-n C-n C-e M-w
C-s S u b C-s C-s C-s ent C-n C-n C-y
C-s < ! - C-s C-s C-s ent C-b C-b C-spc C-f C-f C-f C-f C-f C-f M-w
C-s S h o u l d spc o u C-s C-e C-b C-b C-b C-b C-y
C-s < ! - C-s C-s C-s C-s ent C-a C-d C-spc C-e M-w
C-s < / h t C-s C-s ent C-y
4. (all above)
5. C-s e n d o f h t m C-s C-e C-spc C-x ] C-d
6. C-x u
7. C-s e n d o f h t m C-s C-e < ! - - C-x ] - - >
8. M-% < o l > enter < O l > enter y y y y y y y
(Replaced 7 instances)
```

```
9. diff -u exer1.html exer4.html >exer4.diff

1.5
1. c d enter m k d i r spc j u n k enter
2. c d spc j u n k ent
t o u c h spc h e l l o . c ent
e m a c s spc he l l o . c
3. M-x c o m p i l e ent DOWN
g c c spc - o spc h e l l o spc h e l l o . c ent
4. M-x c o m p i l e ent DOWN
. / h e l l o ent C-x 0 ent C-x C-w h e l l o . o u t ent
5. e m a c s spc h e l l o . o u t C-spc C-x ] M-w C-z
e m a c s spc k e y s 1 . t x t C-x ] C-y

-*- mode: compilation; default-directory: "~/junk/" -*-
Compilation started at Wed Oct  4 22:24:12

./hello
#include <stdio.h>
int
main (void)
{
  char n = '\n';
  char b = '\\';
  char q = '"';
  char const *p = "#include <stdio.h>%cint%cmain (void)%c{%c  char n =
'%cn';%c  char b = '%c%c';%c  char q = '%c';%c  char const *p = %c%s
%c;%c  printf (p, n, n, n, n, b, n, b, b, n, q, n, q, p, q, n, n\
, n, n);%c  return 0;%c}%c";
  printf (p, n, n, n, n, b, n, b, b, n, q, n, q, p, q, n, n, n, n);
  return 0;
}

Compilation finished at Wed Oct  4 22:24:12

1.6
1. C-x b enter
2. ( r a n d o m ) C-j
3. ( s e t q spc x ( r a n d o m ) ) C-j
   822161580090918531
   ( s e t q spc y ( r a n d o m ) ) C-j
   -1749128491421410848
4. ( * spc x spc y) C-j
   494584829784341920
   The result is not correct, it should be negative and a lot bigger.
   The result is just a random number.
5. M-: ( * spc x spc y) C-j
   494584829784341920
6. A number can't actually be randomly generated.
All programs follow some sort of algorithm.
```

There will always be a 'seed' number that every random number starts from.
A pattern is followed to create 'psuedo' random numbers.
7.The two-variable product would be incorrect when the product overflows.
The random returns integers.
I have a 64-bit system, so the maximum number is 2^64-1.
If number is bigger than (2^64-1)/2, the other number has to be 0, 1 ,-1.
Otherwise, the product will overflow
The odds that the number is bigger than (2^64-1)/2 is 1/2.
The odds that the second number is 0, 1, -1 is (3/2^64-1).
Overall, the odds that the product wouldn't overflow is:
(3/(3.7 * 10^19)).
So the odds that it would be incorrect is around 99.9%.