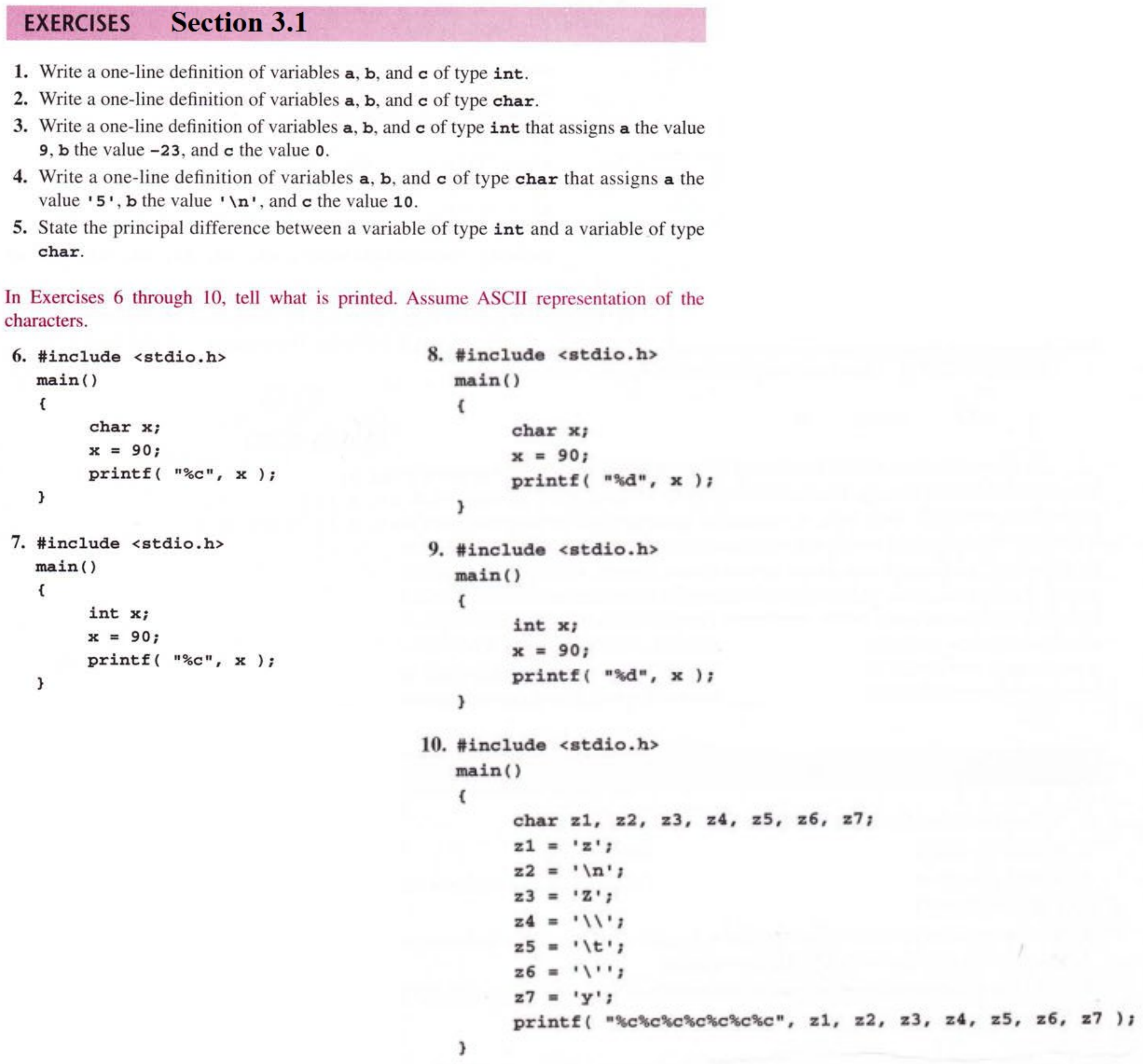
Katrina Viray ID: 1999700

ECE 3331, Dr. Hebert, Fall 2023, HW 3 due Friday 09/08 at 11:59 pm

Problem 1. Section 3.1, Exercise 1.
Problem 2. Section 3.1, Exercise 2.
Problem 3. Section 3.1, Exercise 3.
Problem 4. Section 3.1, Exercise7.
Problem 5. Section 3.1, Exercise 10.
Problem 6. Section 3.2, Exercise 5.
Problem 7. Section 3.3, Exercise 3.
Problem 8. Section 3.3, Exercise 5.
Problem 9. Section 3.3, Exercise 11.
Problem 10. Section 3.4, Exercise 1.
Problem 11. Section 3.5, Exercise1.
Problem 11. Section 3.6, Exercise 3.
Problem 12. Section 3.7, Exercises 6.
Problem 13. Section 3.7, Exercises 7.



EXERCISES Section 3.1

1. Write a one-line definition of variables **a**, **b**, and **c** of type **int**.
2. Write a one-line definition of variables **a**, **b**, and **c** of type **char**.
3. Write a one-line definition of variables **a**, **b**, and **c** of type **int** that assigns **a** the value **9**, **b** the value **-23**, and **c** the value **0**.
4. Write a one-line definition of variables **a**, **b**, and **c** of type **char** that assigns **a** the value **'5'**, **b** the value **'\n'**, and **c** the value **10**.
5. State the principal difference between a variable of type **int** and a variable of type **char**.

In Exercises 6 through 10, tell what is printed. Assume ASCII representation of the characters.

6.
```
#include <stdio.h>
main()
{
      char x;
      x = 90;
      printf( "%c", x );
}
```

7.
```
#include <stdio.h>
main()
{
      int x;
      x = 90;
      printf( "%c", x );
}
```

8.
```
#include <stdio.h>
main()
{
      char x;
      x = 90;
      printf( "%d", x );
}
```

9.
```
#include <stdio.h>
main()
{
      int x;
      x = 90;
      printf( "%d", x );
}
```

10.
```
#include <stdio.h>
main()
{
      char z1, z2, z3, z4, z5, z6, z7;
      z1 = 'z';
      z2 = '\n';
      z3 = 'Z';
      z4 = '\\';
      z5 = '\t';
      z6 = '\'';
      z7 = 'y';
      printf( "%c%c%c%c%c%c%c", z1, z2, z3, z4, z5, z6, z7 );
}
```

EXERCISES Section 3.2

In Exercises 1 through 4, express each number in exponential notation as used in C.

1. 399481.772 **2.** –9987768791.19002

3. .00000000022815 **4.** –.00000005983

5. What is printed? (Assume ASCII representation.)

```
#include <stdio.h>
main()
{
      int i;
      char c;
      float x;
      i = 8;
      c = '\n';
      x = 42.4907;
      printf( "i = %d%c", i, c );
      printf( "%c\tc = %d%c", c, c, c );
      printf( "x = %e\tx = %f", x, x );
}
```

6. Write and run a program that prints the value of each constant in Figure 3.6.

EXERCISES Section 3.3

Give the value of the variable on the left side of the assignment operator in each of Exercises 1 through 13. The statements are executed sequentially. Assume that just before each statement, we have

```
int i, j, k;
float x, y, z;
i = 3;
j = 5;
x = 4.3;
y = 58.209;
```

1. `k = j * i;` **2.** `k = j / i;`

3. `z = x / i;` **4.** `k = x / i;`

5. `z = y / x;` **6.** `k = y / x;`

7. `i = 3 + 2 * j;` **8.** `k = j % i;`

9. `k = i % j;` **10.** `k = j % i * 4;`

11. `i += j;` **12.** `j -= x;`

13. `i %= j;`

EXERCISES Section 3.4

1. Assuming that the value of **x** is 21, the value of **y** is 4, the value of **z** is 8, the value of **c** is **'A'**, and the value of **d** is **'H'**, fill in the missing entries in the following table. Also, give the values of all the variables. Assume that ASCII representation is being used. In some cases, you may need to refer to the ASCII table in Appendix A.

Expression	*Value*
`x + y >= z`	1
`y == x - 2 * z - 1`	
`6 * x != x`	
`c > d`	0
`x = y == 4`	
`(x = y) == 4`	
`(x = 1) == 1`	
`2 * c > d`	

2. Assuming that the value of **x** is 11, the value of **y** is 6, the value of **z** is 1, the value of **c** is **'k'**, and the value of **d** is **'y'**, fill in the missing entries in the following table. Also, give the values of all the variables. Assume that ASCII representation is used. In some cases, you may need to refer to the ASCII table in Appendix A.

<table>
<tr><th><i>Expression</i></th><th><i>Value</i></th></tr>
<tr><td><code>x > 9 && y != 3</code></td><td>1</td></tr>
<tr><td><code>x == 5 || y != 3</code></td><td></td></tr>
<tr><td><code>!(x > 14)</code></td><td></td></tr>
<tr><td><code>!(x > 9 && y != 23)</code></td><td></td></tr>
<tr><td><code>x <= 1 && y == 6 || z < 4</code></td><td></td></tr>
<tr><td><code>c >= 'a' && c <= 'z'</code></td><td></td></tr>
<tr><td><code>c >= 'A' || c <= 'Z'</code></td><td></td></tr>
<tr><td><code>c != d && c != '\n'</code></td><td></td></tr>
<tr><td><code>5 && y != 8 || 0</code></td><td></td></tr>
<tr><td><code>x >= y >= z</code></td><td></td></tr>
</table>

3. What is the likely logical error?

```
if ( code == 1 & flag == 0 )
      printf( "OK\n" );
```

4. What is the likely logical error?

```
if ( code == 1 | flag == 0 )
      printf( "ERROR\n" );
```

EXERCISE Section 3.5

1. Can we replace the lines

```
if ( x > max )
      max = x;
if ( x < min )
      min = x;
```

by the lines

```
if ( x > max )
      max = x;
else
      min = x;
```

EXERCISES Section 3.6

1. What is printed?

```
for ( i = 1; i <= 5; printf( "%d\n", i ) )
      i++;
```

2. What is printed?

```
for ( i = 1; i <= 5; i++ ) {
      printf( "%d\n", i );
      i += 2;
}
```

3. What is printed?

```
for ( i = 1; i <= 5; i++ ) ;
      printf( "%d\n", i );
printf( "%d\n", i );
```

4. What is the error?

```
for ( i = 1, i <= 5, i++ )
      printf( "%d\n", i );
```

5. Using a **for** loop, write a program that computes and prints the value of the sum 2 + 4 + . . . + 100.

6. On one system, when the program

```
#include <stdio.h>

main()
{
          float x;
          for ( x = 0; x <= 1.0; x += 0.1 )
                    printf( "%.1f ", x );
}
```

is run, the output is

```
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

(The format descriptor **%.1f** causes one digit after the decimal to be printed.) Where's the missing **1.0**?

EXERCISES Section 3.7

Give the value of **i** and **k** after each statement is executed in Exercises 1 through 5. The statements are *not* executed sequentially. Assume that just before each statement, we have

```
int i, j, k;
i = 3;
k = 0;
```

1. `k = ++i;`　　　2. `k = i++;`

3. `k = --i;`　　　4. `k = i--;`

5. `i = j = k--;`

6. Assuming that ASCII representation is used, the value of **c** is **'k'**, and the value of **d** is **'y'**, find the value of the expression

```
--c == 'j' && d++ == 'y'
```

7. What is printed?

```
int x = 7;
if ( x-- < 2 )
   printf( "%d", x );
else if ( x-- < 4 )
   printf( "%d", 2 * x );
else if ( x-- < 6 )
   printf( "%d", 3 * x );
else
   printf( "%d", 4 * x );
```

8. Do the two code fragments always produce the same output given the same input? Explain.

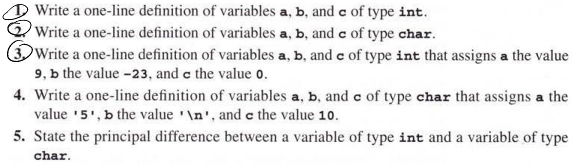
```
if ( code++ <= 3 )                  if ( code++ <= 3 )
   printf( "%d\n", code );             printf( "%d\n", code );
else if ( code++ <= 5 )             if ( code++ <= 5 )
   printf( "%d\n", code );             printf( "%d\n", code );
else if ( code++ <= 7 )             if ( code++ <= 7 )
   printf( "%d\n", code );             printf( "%d\n", code );
```

If the fragments are logically the same, is one form preferable to the other? Explain.

9. How is the expression

```
x----y
```

parsed? Is it a legal expression?

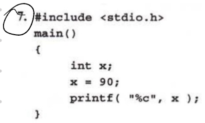


1. Write a one-line definition of variables **a**, **b**, and **c** of type **int**.
2. Write a one-line definition of variables **a**, **b**, and **c** of type **char**.
3. Write a one-line definition of variables **a**, **b**, and **c** of type **int** that assigns **a** the value **9**, **b** the value **-23**, and **c** the value **0**.
4. Write a one-line definition of variables **a**, **b**, and **c** of type **char** that assigns **a** the value **'5'**, **b** the value **'\n'**, and **c** the value **10**.
5. State the principal difference between a variable of type **int** and a variable of type **char**.

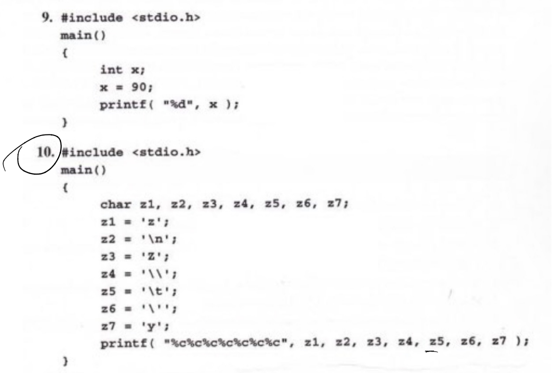
1. int a, b, c;

2. char a, b, c;

3. int a = 9, b = -23, c = 0



7.
```
#include <stdio.h>
main()
{
      int x;
      x = 90;
      printf( "%c", x );
}
```



9.
```
#include <stdio.h>
main()
{
      int x;
      x = 90;
      printf( "%d", x );
}
```

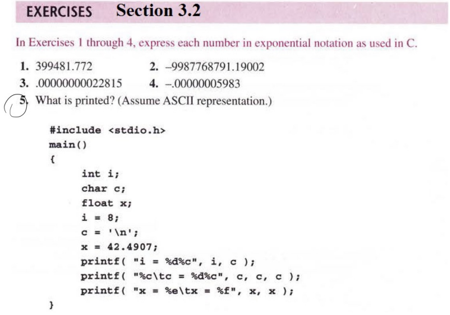
10.
```
#include <stdio.h>
main()
{
      char z1, z2, z3, z4, z5, z6, z7;
      z1 = 'z';
      z2 = '\n';
      z3 = 'Z';
      z4 = '\\';
      z5 = '\t';
      z6 = '\'';
      z7 = 'y';
      printf( "%c%c%c%c%c%c%c", z1, z2, z3, z4, z5, z6, z7 );
}
```

What is printed?

4. Z

5. z

Z\   'y



EXERCISES Section 3.2

In Exercises 1 through 4, express each number in exponential notation as used in C.

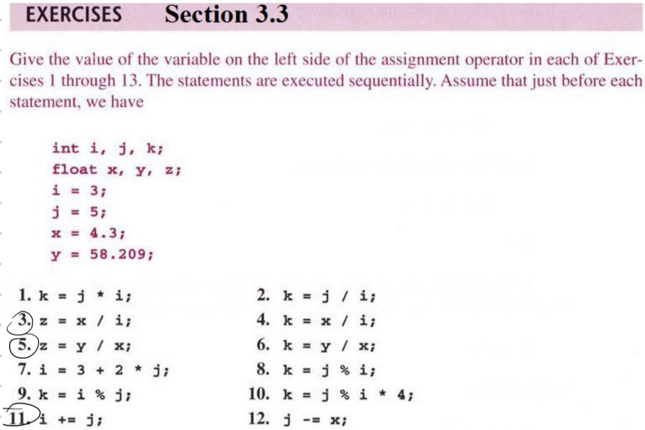
1. 399481.772
2. −9987768791.19002
3. .00000000022815
4. −.00000005983
5. What is printed? (Assume ASCII representation.)

```
#include <stdio.h>
main()
{
     int i;
     char c;
     float x;
     i = 8;
     c = '\n';
     x = 42.4907;
     printf( "i = %d%c", i, c );
     printf( "%c\tc = %d%c", c, c, c );
     printf( "x = %e\tx = %f", x, x );
}
```

6. i = 8

c = 10

x = 4.24907e+01 x=42.490700



EXERCISES Section 3.3

Give the value of the variable on the left side of the assignment operator in each of Exercises 1 through 13. The statements are executed sequentially. Assume that just before each statement, we have

```
int i, j, k;
float x, y, z;
i = 3;
j = 5;
x = 4.3;
y = 58.209;
```

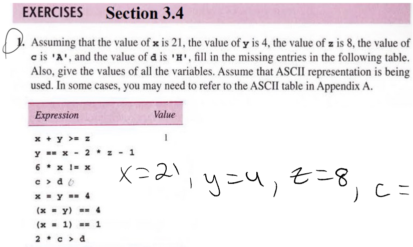
1. `k = j * i;`
2. `k = j / i;`
3. `z = x / i;`
4. `k = x / i;`
5. `z = y / x;`
6. `k = y / x;`
7. `i = 3 + 2 * j;`
8. `k = j % i;`
9. `k = i % j;`
10. `k = j % i * 4;`
11. `i += j;`
12. `j -= x;`

```
   1.43
3 )4.300
 - 3
   1.3
 - 1.2
     10
   - 9
     10
```

7. 1.433333

8. 13.536978

9. 8



EXERCISES Section 3.4

1. Assuming that the value of **x** is 21, the value of **y** is 4, the value of **z** is 8, the value of **c** is **'A'**, and the value of **d** is **'H'**, fill in the missing entries in the following table. Also, give the values of all the variables. Assume that ASCII representation is being used. In some cases, you may need to refer to the ASCII table in Appendix A.

Expression	*Value*
`x + y >= z`	1
`y == x - 2 * z - 1`	
`6 * x != x`	
`c > d`	0
`x = y == 4`	
`(x = y) == 4`	
`(x = 1) == 1`	
`2 * c > d`	

x=21, y=4, z=8, c='A', d='H'

10.

expression	Value	values of variables
x + y >= z	1	unchanged
y == x-2*z-1	1	unchanged
6* x != x	1	unchanged
c > d	0	unchanged
x = y == 4	1	x = 1
(x = y) == 4	1	x = 4
(x = 1) == 1	1	~~x~~ = 1
2 * c > d	1	unchanged

4 = 21 - 2*8 - 1 → 4 = 20 - 16 → ✓

6 · 21 != 21

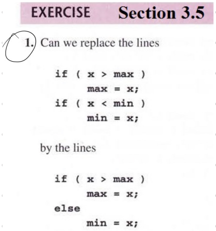
A > H ✗

4 == 4

4 == 4

x == 1

2 · 'A' > 'H'



EXERCISE Section 3.5

1. Can we replace the lines

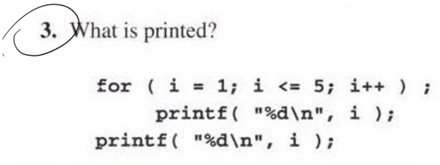
```
if ( x > max )
      max = x;
if ( x < min )
      min = x;
```

by the lines

```
if ( x > max )
      max = x;
else
      min = x;
```

11. No. If x is not less than min, then the min is still replaced by x, which we don't want to happen.

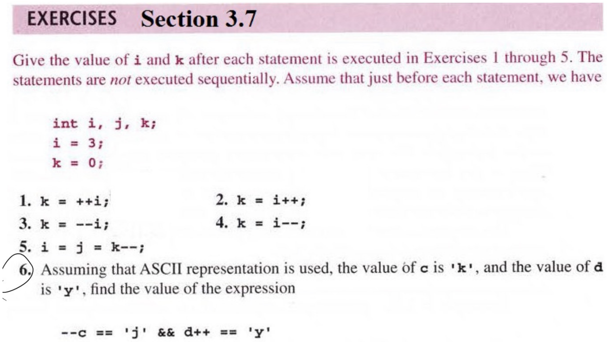
Section 3.6



3. What is printed?

```
for ( i = 1; i <= 5; i++ ) ;
     printf( "%d\n", i );
printf( "%d\n", i );
```

11. 1
2
3
4
5



EXERCISES Section 3.7

Give the value of **i** and **k** after each statement is executed in Exercises 1 through 5. The statements are *not* executed sequentially. Assume that just before each statement, we have

```
int i, j, k;
i = 3;
k = 0;
```

1. `k = ++i;` **2.** `k = i++;`

3. `k = --i;` **4.** `k = i--;`

5. `i = j = k--;`

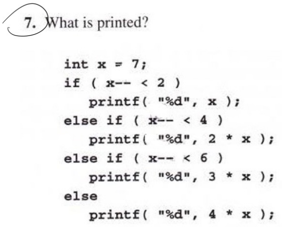
6. Assuming that ASCII representation is used, the value of **c** is **'k'**, and the value of **d** is **'y'**, find the value of the expression

```
--c == 'j' && d++ == 'y'
```

-- 107 == 106 && ~~100~~ 121 ++ == 121

106 == 106 && 121+1 == 121

12. 1



7. What is printed?

```
int x = 7;
if ( x-- < 2 )
   printf( "%d", x );
else if ( x-- < 4 )
   printf( "%d", 2 * x );
else if ( x-- < 6 )
   printf( "%d", 3 * x );
else
   printf( "%d", 4 * x );
```

13. 12