Points: 55/50

Problem 1: (7.5 points)

Show the output of the following code statements and briefly describe why the output looks like that.

1) `printf("%u", -1);`

Output: 4294967295
Since we are printing an unsigned number, negatives cannot exist. So, the compiler goes back to the highest number that an unsigned integer can hold.

2) `printf("%d", '0');`

Output: 48
This is the ASCII number for the character '0'.

3) `printf("%E", 123.45);`

Output: 1.234500E+02
The %E stands for exponential form, thus the number is put into exponential form.

4) `printf("%d", +1);`

Output: 1
The sign is not printed because there is not a plus sign flag in front of the conversion specifier telling the compiler to print the sign.

5) `printf("12%2d45", 3);`

Output: 12 345
The 12 in front of the conversion specifier is printed as normal. Then, the integer 3 is printed in a field width of 2 and then the 45 is printed immediately after.

Problem 2:

Show/describe the error in the following code statements write the correct version.

1)
```
int x = -1;
printf("%u", x);
```

Error: %u stands for unsigned integer. %d would print the proper negative number.
Correct code:
int x = -1;
printf("%d", x);

2)
```
float var = 2.0;
scanf("%f", var);
```

Points: 55/50

Error: scanf is used
instead of printf.
Correct code:
float var = 2.0;
printf(“%f”, var);

(10.5 points)

3) int x = 5;
printf("&d", x);

Error: The address operator & is used instead of the conversion specifier %.
Correct code:
int x = 5;
printf(“%d”, x);

Problem 3: (15 points)

Create a printf statement for each of the following outputs. You must use flags, field widths, and precision. **Note: '_' is used to represent space.**

1) +321
printf(“+%d”, 321);
2) 0321
printf("%04d", 321);
3) _321
printf(“%4d”, 321);
4) 321_
printf(“-%4d”, 321);
5) 3.210
printf(“%.3f”, 3.21);

Points: 55/50

Problem 4: (17 points)

Design a small program that will keep track of a sample bank account balance. The starting balance is $162.54.

Note:

- **When printing the dollar amount, please include the '$' symbol and display 2 points of decimal precision.**
- **Provide your source code (.c) file in a separate file**
- **Provide snapshots of your running code. Attach the outputs in a word or PDF file.**

Instructions:

- Show the user the current account balance.
- A prompt will ask the user to deposit (denoted with the character '+') to the account or withdraw (denoted with the character '-') from the account.
- A prompt will ask for the amount of money.
- Perform the addition or subtraction and update account balance variable.
- Show the new balance of the account to the user.

Example (deposit):

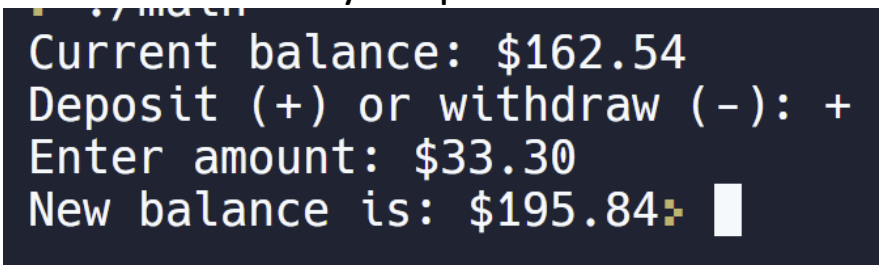
```
Current balance: $162.54
Deposit (+) or withdraw (-): +
Enter amount: $11.46
New balance is: $174.00
```

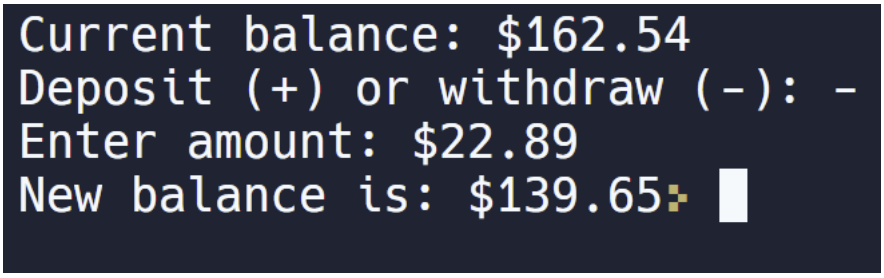
Example (withdraw):

```
Current balance: $162.54
Deposit (+) or withdraw (-): -
Enter amount: $3.99
New balance is: $158.55
```

Points: 55/50

Screenshots of my output:





Points: 55/50

Problem 5 (bonus): (5 points)

Write a C program that asks the user to enter four integers, representing the weights in kilograms and pounds. After scanning the values the program should print a formatted table. Your program should provide a table with **exactly the same format**.

You should print two entered kilogram values in the first column, and two corresponding weights in pounds in the second column (1 kg = 2.2 lb). Two entered pound weights in the third column, and two corresponding weights in kilograms in the fourth column.

The width of the first column is 9 digits, second column – 13 digits, third column - 9 digits, fourth 13 digits. There are two spaces, vertical line and another two spaces between the second and the third columns.

See the examples below. Run your program and show the output **with the same input values.**

```
Enter two weights in kilograms:
10
1000
Enter two weights in pounds:
10000
300
-----------------------------------------------------
10        |         22.0  |  10000    |      4545.45
     1000|      2200.00  |  300      |       136.364
-----------------------------------------------------
```

```
Enter two weights in kilograms:
9
99999
Enter two weights in pounds:
7777777
7
-----------------------------------------------------
9         |         19.8  |  7777777  |   3535353.25
    99999|    219997.80  |  7        |         3.182
-----------------------------------------------------
```

Points: 55/50

Screenshots of my output:

```
Enter two weights in kilograms:
10
1000
Enter two weights in pounds:
10000
300
-----------------------------------------------------
10       |         22.0  |  10000    |      4545.45
     1000|      2200.00  |  300      |       136.364
-----------------------------------------------------
>
```

```
Enter two weights in kilograms:
20
500
Enter two weights in pounds:
50093
456
-----------------------------------------------------
20       |         44.0  |  50093    |     22769.54
      500|      1100.00  |  456      |      207.273
-----------------------------------------------------
>
```