1. What is the output generated by the following code?

```fortran
i = 0
j = -2
k = 2
do 10 i=j,k
   print *, i,j,k
   j = j - i
   k = k + i
10 continue
```

---

`i` takes on values -2,-1,0,1,2 (the i=0 statement does nothing)

Output:

```
-2, -2, 2 then j = -2 - (-2) = 0 and k = 2 + (-2) = 0
-1, 0, 0 then j = 0 - (-1) = 1 and k = 0 + (-1) = -1
0, 1, -1 then j = 0 - 0 = 1 and k = -1 + 0 = -1
1, 1, -1 then j = 1 - 1 = 0 and k = -1 + 1 = 0
2, 0, 0 then we don’t care about j and k (though they are changed)
```

2. What is the output generated by the following code?

```fortran
do 100 i=1,3
do 90 j=0,5,2
do 80 k=i,j
   print *, i+j+k
80    continue
90    continue
100   continue
```

Answer on back.

(continued on back)
when i=1: k goes from 1 to 0 (no output), 1 to 2, and 1 to 4
when i=2: k goes from 2 to 0 (no output), 2 to 2, and 2 to 4
when i=3: k goes from 3 to 0 (no output), 3 to 2 (no output), and 3 to 4

Output: +----+ i, j, k
        | 4   | 1 2 1
        | 5   | 1 2 2
        | 6   | 1 4 1
        | 7   | 1 4 2
        | 8   | 1 4 3
        | 9   | 1 4 4
        | 6   | 2 2 2
        | 8   | 2 4 2
        | 9   | 2 4 3
        | 10  | 2 4 4
        | 10  | 3 4 3
        | 11  | 3 4 4

3. What is the output generated by the following code?

    do 10 x=1.25,5.5
       print *, 2.0 * x
    10 continue

Output: +----+ x
        | 2.5  | 1.25
        | 4.5  | 2.25
        | 6.5  | 3.25
        | 8.5  | 4.25
        | 10.5 | 5.25

4. Using a DO-loop, write code that will read a value for the integer variable n and then print all multiples of n between 1 and 1000. You could use the MOD() function to solve this problem, but that would be inefficient. Try to come up with a better way.

    read *, n
    do 10 i=1,1000/n
       print *,i*n
    10 continue
5. Using a DO-loop, write code that will read a value for the integer variable \( n \), then read \( n \) real values and print their sum. For example, if the user entered 3 for \( n \), the code should read in three values, and then print the sum of those values.

```fortran
read *, n
sum = 0.0
do 10 i=1,n
   read *,x
   sum = sum + x
10 continue
print *,sum
```

6. Using a DO-loop, write code that will read a value for the integer variable \( n \) and then print \( n \) equally-spaced points in the range from \(-5.0\) to \(5.0\), including the end points, together with the value of \( e^{-x} \cos(x) \) evaluated at these points. (You can assume that \( n \) will always be at least two.) As an example, if the user entered 3 for \( n \), the output would be:

```
-5. 42.099205
 0. 1.
 5. 0.00191130082
```

You will need to do some calculating of variables outside of the DO-loop. For the sake of this homework, any variables other than \( i, j, k, \) and \( n \) will be considered real, so you don’t need to explicitly declare anything. Just write the relevant code to accomplish the task.

```fortran
read *, n
delta = 10.0/(n-1)
do 10 i=0,n-1
   x = i*delta - 5.0
   print *, x, exp(-x)*cos(x)
10 continue
```

Note that you don’t absolutely have to calculate \( \delta \) outside the DO-loop, but it is more efficient to. One could write:

```fortran
read *, n
do 10 i=0,n-1
   x = i*10.0/(n-1) - 5.0
   print *, x, exp(-x)*cos(x)
10 continue
```

In the previous problem you do have to initialize \( \text{sum} \) outside the loop.