John Heinlein   **Lab 8 - 4/8/19**

1. Record the addresses of the variables.
   i1*: 0x0x7ffdbaf94a64* i2*: 0x0x7ffdbaf94a60*
   d1*: 0x0x7ffdbaf94a58* d2*: 0x0x7ffdbaf94a50*

2. Declare 4 pointer variables (intptr1, intptr2, dubptr1, dubptr2), one for each of the above variables and record the addresses of these variables.
   Address of intptr1: *0x0x7ffdbaf94a48*
   Address of intptr2: *0x0x7ffdbaf94a40*
   Address of dubptr1: *0x0x7ffdbaf94a38*
   Address of dubptr2: *0x0x7ffdbaf94a30*

3. Assign the address of i1 to intptr1, the address of d1 to dubptr1 and so on. Record the data values stored in the pointer variables.
   Value of intptr1: 0x7ffc0947b1d4
   Value of intptr2: 0x7ffc0947b1d0
   Value of dubptr1: 0x7ffc0947b1c8
   Value of dubptr2: 0x7ffc0947b1c0

4. Assign intptr2 to intptr1 and record the value of intptr1. Assign intptr1 to dubptr1 and record what happens. Use typecasting to cast the type of intptr1 to type (double *) and assign this to dubptr1 and record the value in dubptr1.
   Value of intptr1: *0x7ffc0947b1d0*
   Assignment of intptr1 to dubptr1: *0x7ffc0948b1d0 (with warning given by cc: “assignment from incompatible pointer type”)*
   Value of dubptr1 after typecasting: *0x7ffc0948b1d0*

5. Assign the value NULL to intptr1 and record the value that is output for intptr1.
   Value of intptr1: *(nil)*

6. Dereference the pointer intptr2 and print the result. Try to dereference the pointer intptr1 (which is set to NULL) and see what happens. If this causes a problem with the program, record the problem and remove the code.
   Value of *intptr2: *2*
   Value of *intptr1: *Segfault*

7. Assign the value 100 to * intptr2 and record the value of both i1 and i2.
   Value of i1: *1*
   Value of i2: *100*

8. For intptr2 and dubptr2, record the value of and the dereferenced value of the pointer + 1 and the pointer – 1.
   Value of (intptr2 + 1): *0x7ffc0947b1d4*
   Value of *(intptr2 + 1): *1*
   Value of (intptr2 – 1): *0x7ffc0947b1cc*
   Value of *(intptr2 – 1): *1072798105*

   Value of (dubptr2 + 1): *0x7ffc0947b1c8*
   Value of *(dubptr2 + 1): *1.100000*
   Value of (dubptr2 – 1): *0x7ffc0947b1b8*
   Value of *(dubptr2 – 1): *0.000000*

9. Set intptr1 to the address of i1. Record the answers to the following questions:
    intptr1 == intptr2? *No*
   *intptr1 == *intptr2? *No*
   Now set intptr1 to the address of i2 and record the answers.
    intptr1 == intptr2? *Yes*
   *intptr1 == *intptr2? *Yes*

10. The malloc function assigns new memory to a pointer variable. It returns type void * so you must typecast it to the correct type. Declare a new pointer to type double (ptr) and assign a block of memory to this pointer using malloc (sizeof (double)).
    Using the pointer, assign the value of 3.1416 to this block of memory. Record the following information:
    Value of ptr: *0x20f3010*
    Value of *ptr: *3.141600*

11. To deallocate the dynamic memory of a pointer, we use the free () function. This function requires us to typecast our pointer to type void *. The syntax is: free ((void *) ptr); Deallocate the memory for ptr and reallocate it again.
    Is the value of ptr the same as it was in question 10?
    The value of ptr is the same, but the data stored there has been zeroed