Copy this code into a file and run it so you can see what prints for the address of \texttt{a} is from each function and the address of \texttt{ptr}.

Then, show what is printed to the screen after running the program:
/* pointers.c */
#include <stdio.h>

int main (void) {
    int count1 = 10;
    int *ptr1 = &count1;

    int count2 = 20;
    int *ptr2 = &count2;

    printf ("ptr1 is %i, ptr2 is %i \n", *ptr1, *ptr2);
    ptr1 = ptr2;
    printf ("ptr1 is %i, ptr2 is %i \n", *ptr1, *ptr2);
    return 0;
}

Show what is printed to the screen after running the program:

____________________________________________________________________________________
____________________________________
_______________________________________________________________________________________________________________________
#3 – pass by reference example

```c
#include <stdio.h>

void modify(int p, int *q, int *r) {
    printf("\n1. p is %i, q is %i, r is %i \n", p, *q, *r);
    p = 27;
    *q = 27;
    *r = 27;
}

int main (void) {
    int a = 1;
    int b = 1;
    int c = 1;
    int *x = &c;

    modify(a, &b, x);
    printf("2. a is %i, b is %i, c is %i, x is %i \n\n", a, b, c, *x);

    return 0;
}
```

Show what is printed to the screen after running the program:

```
-----------------------------------------------
-----------------------------------------------
```

Which values are different after `main`'s print statement and why?
What are the answers from the 4 questions above in the comments?

______________________________

______________________________

______________________________

______________________________

Show what is printed to the screen after running the program:

______________________________

______________________________
#5 – more pointers – practice with tracing through code on paper

```c
// tracing.c
#include <stdio.h>

int main (void) {
    int values[10] = { -1, 14, -24, 6, 9, 2, -3, 4, 7, 3 };
    char word[32] = "The semester is just beginning!";
    int i;

    printf("\n\n\n");
    int *ptr1 = values;
    printf ("ptr1 = %i \n", *ptr1);

    int *ptr2 = ptr1 + 3;
    printf ("ptr2 = %i \n", *ptr2);

    char *ptr3;
    ptr3 = word;
    printf ("*ptr3 = %c \n", *ptr3);
    printf ("*(ptr3 + 4) = %c \n", *(ptr3 + 4));

    printf ("Letters: ");
    char *ptr4 = word;
    for (i=0; i<8; i++) {
        printf ("%c", *ptr4);
        ptr4 += 4;
    }

    printf ("\n%s \n", word);
    printf("\n\n");
    return 0;
}
```

Trace through the code on paper (before copying it into a file and running it) and show what is printed to the screen after running the program:
#6 – review of static

```c
/* static.c */
#include <stdio.h>

int tryToModify (int a) {
    static int b = 5;
    a = a + 5;
    b = b + 5;
    printf("a is %i, b is %i \n", a, b);
}

int main (void) {
    int x = 20;
    printf("\n\n");
    tryToModify (x);
    tryToModify (x);
    tryToModify (x);
    printf("\n\n");
    return 0;
}
```

Show what is printed to the screen after running the program:

_______________________________________________________________________________________________________________________

_______________________________________________________________________________________________________________________

_______________________________________________________________________________________________________________________
#7 – review of malloc

```c
// malloc.c
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    int a = 10;
    int b = 12;
    int *base = NULL;
    base = (int *) malloc(sizeof(int) * 100);

    printf("a is %p \n", &a);
    printf("b is %p \n", &b);
    printf("base is %p \n", base);

    return 0;
}
```

Copy this code into a file and run it so you can see what prints for the addresses of the above variables.

Then, show what is printed to the screen after running the program:

```
Some output will be displayed here.
```

What do you notice about the three addresses, other than they are some hex number?
#8 – review of heap

```c
// heap.c

#include <stdio.h>

int main (void)
{
    int y;
    int *ptr;
    static int a;

    ptr = &y;
    *ptr = 99;

    printf ("y is %i \n ptr is %i \n address of ptr is %p \n", y, *ptr, &ptr);

    ptr = &a;
    printf ("the address of a is %p \n", ptr);

    return 0;
}
```

Copy this code into a file and run it so you can see what prints for the addresses of the above variables.

Then, show what is printed to the screen after running the program:

```
_______________________________________________________________________________________________________________________
_______________________________________________________________________________________________________________________
_______________________________________________________________________________________________________________________
_______________________________________________________________________________________________________________________
```

What do you notice about the two addresses, other than they are some hex number?