CS270 – Dynamic memory   Name____________________________________

1) What does the following code print:

```
#include <stdio.h>
int main() {
        int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
        int *ptr = &arr[0];
        while (ptr <= &arr[3]) { // pointer comparison
                printf("%d ", *ptr);
                ptr++;
        }
        printf("\n");
}
```

0 1 2 3

2) Consider the following program, where are i, j and k are stored in memory?

```
int i;            // data segment
int main()
{
    int j;        // stack
    int *k = (int *) malloc (sizeof(int)); // heap
}
```

3) What is the output?

```
# include<stdio.h>
# include<stdlib.h>

void fun(int *a)
{
    a = (int*)malloc(sizeof(int)); // a points to malloc allocated memory
}

int main()
{
    int *p;         // allocate uninitialized memory on the stack for an int *
    fun(p);         // pass a copy of p to fun()
    *p = 6;         // dereference uninitialized/garbage value and store 6 at
                    // that location, causes a segfault
    printf("%dn",*p);
    return(0);
}
```

4) What is wrong with the following code?

```
#include<stdio.h>
int main()
{
    int *p = (int *)malloc(sizeof(int));
    p = NULL;       // p now points to NULL, and address of allocated memory is
                    // lost
    free(p);        // no memory freed which causes a memory leak because p no
                    // longer points to the allocated memory
}
```

5) Which of the following three functions are likely to cause problems with pointers?

```
int * g (void)
{
  int x= 10;
  return (&x);      // returning a pointed to a local/stack variable that is
                    // not valid after return from g();
}

int * g (void)
{
  int * px;         // allocate uninitialized memory on the stack for an int *
  *px= 10;          // dereference uninitialized/garbage value and store 6 at
                    // that location, causes a segfault
  return px;
}

int *g (void)
{
  int *px;          // allocate uninitialized memory on the stack for an int *
  px = (int *) malloc(sizeof(int));       // give px address of a valid
                                          // memory location
  *px= 10;          // dereference valid memory location and store value
  return px;        // return pointer to heap allocated memory that persists
                    // after function returns
}
```

6) What does the following code print?

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r = 3, c = 4;
    int *arr = (int *)malloc(r * c * sizeof(int));// malloc memory for 2d array

    int i, j, count = 0;
    for (i = 0; i <  r; i++)
      for (j = 0; j < c; j++)
         *(arr + i*c + j) = ++count; // store 1 - 12 in the 2d array

    for (i = 0; i <  r; i++)
      for (j = 0; j < c; j++)
         printf("%d ", *(arr + i*c + j)); // print 1 – 12 from the 2d array

   /* Code for further processing and free the
      dynamically allocated memory */

   return 0;
}
```

1 2 3 4 5 6 7 8 9 10 11 12