HW: C Programming, Part 2 Name Umberto Fontana
IC221, Spring AY23 100 points total

1. (10 points) Complete the program below such that it produces the expected output.

```
struct pair{
  int left;
  int right;
};

int main(int argc, char * argv[]){
  struct pair p;
  struct pair *q;

  q = &p;

  p.left=20;
  p.right=10;
```

 // Printing the pair using p and q?

```
  printf("p: (%d,%d)\n", /* What goes here? */ );

  printf("q: (%d,%d)\n", /* What goes here? */);
```

```
printf("p: (%d,%d)\n", p.left, p.right )
printf("q: (%d,%d)\n", q->left, q->right);
```

2. (10 points) Convert the following declaration into an equivalent using *array* notation.

```
char s1[] = "Beat Army!";

char s2[] = { /* what goes here? */ };
```

```
char s2[] = {'B','e','a','t',' ','A','r','m','y','!','\0'};
```

3. (10 points) What is the output of running the following code snippet below?

```
char s[] = "Beat Army\0Crash Airforce\0";

printf("1: %s\n",s);
printf("2: %s\n",s+17);
```

```
Beat Army
irforce
```

4. (10 points) Complete the program below to copy the contents of s1 to s2.

```
int main(){
  char s1[] = "I love IC221!";
  char s2[strlen(s1)];

  int i;
  for(i=0;i<strlen(s1);i++){
     s2[i] = s1[i];
  }

}
```

5. (10 points) Look up the following string library functions using the manual pages. Provide a short description of each:

`strcpy()`	Copies the string pointed to by src into a string at the buffer pointed to by dst. The programmer is responsible for allocating a destination buffer of length strlen(src)+1. If the buffer is not large enough, the behavior is undefined. Returns dst.
`strncpy()`	Fills a fixed-size buffer with non-null bytes from a string, padding with null bytes as needed. If the destination buffer, limited by its size, isn't large enough to hold the copy, the resulting character sequence is truncated. Returns dst.
`strcat()`	This function concatenates the string pointed to by src, after the string pointed to by dst (overwriting its terminating null byte). The programmer is responsible for allocating a destination buffer large enough (strlen(dst) + strlen(src) +1). If the destination buffer is not large enough, the behavior is undefined. Returns dst.
`strfry()`	Randomizes the contents of string by randomly swapping characters in the string. The result is an anagram of string. Returns a pointer to the randomized string.
`strchr()`	Locates character c in a string s. Returns a pointer to the first occurrence of the character c in the string s, or NULL if not found. Does not work with wide or multibyte characters.

6. (10 points) Consider the following program. What is its output?

```
int main(){
  int darray[][4] = {{1, 9, 8, 4},
                     {1, 8, 9, 4},
                     {2, 0, 1, 7},
                     {3, 4, 5, 8}};

  int * p = darray[1];

  printf("%d\n", p[2]);
}
```

```
9
```

7. (10 points) Explain why the following type declaration for an array of strings actually represents a 'double array.'

```
char * my_string[];
```

A string is an array of characters. An array of arrays of characters is a double array.

8. (10 points) Complete the following memory diagram for the `argv[]` array for the following command and arguments:

```
$ ./cmd go navy
```

```
          .---.
argv[0]-> | .-+--> "./cmd"
          |---|
argv[1]-> | .-+--> "go"
argv[2]->   .-+--> "navy"
argv[3]->   .-+--> "\0"           .
```

9. (10 points) Explain why the following `for` loop iterates over the `argv` array.
(Compile and run the program if it helps you)

```
int main(int argc, char * argv[]){

  char ** curarg;
  int i=0;

  for( curarg=argv; *curarg ; curarg++){
    printf("argv[%d] = %s\n", i++, *curarg);
  }

}
```

This for loop iterates through argv:

- The initial value of curarg, the iterator, is set to the first value of argv. Since argv is an array of strings, which makes it a double array, curarg is a double pointer and gets pointed at the start of the argv array (which is the first string of the command line arguments).
- It then iterates until the iterator, curarg, gets a value of NULL (* curarg). This works because the command line arguments array is NULL terminated, like strings.

10. (10 points) The program below checks if each of the command line arguments is a number. Complete the program by filling in an error-checked use of `sscanf()`.

```
int main( int argc, char *argv[]){
char ** curarg;
int i=0;
int test;

 for( curarg=argv; *curarg ; curarg++){

   // Use sscanf() to perform a number/integer check

   if(sscanf(*curarg, “%d”, &test) != 0) /*Check passes*/

     printf("argv[%d] = %s (is a number!)\n", i++, *curarg);
   else
     printf("argv[%d] = %s (is *NOT* a number!)\n", i++, *curarg);
 }

}
```