HW 4

Mike Hanling

8 FEB 2017

Instructions

- You must turn in a sheet of paper that is neatly typed or written answering the questions below. (You are strongly encouraged to type your homework.)
- This homework is graded out of 110 points. Point values are associated to each question.

Questions

1. (10 points) Complete the program below such that the program 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? */);
```

```
p:  p.left, p.right
q:  q->left, q->right
```

2. (10 points) Convert the following string deceleration into a similar array deceleration.

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

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

'B', 'e', 'a', 't', ' ', 'A', 'r', 'm', 'y'

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);
```

```
1: Beat Army
2: irforce
```

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

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

  for( /* ????  */){
    /* ????? */
  }

}
```

```
char s2[(int)sizeof(s1)];

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

5. (10 points) Look up the following string library functions using the man page for `string.h` and provide a short description of each:

   (a) `strcpy()`

   Copies the second argument into the first argument.

   (b) `strncpy()`

   Copies n characters of the second argument into the first argument.

   (c) `strcat()`

   Concatenates the two arguments together.

   (d) `strfry()`

Turns the input into an anagram of itself.

(e) `strchr()`

Returns a pointer to the first occurance of the second argument in the string. NULL if not not found.

6. (10 points) Consider the following program, what is its output? Provide a short memory diagram to explain.

```
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]);
}
```

Output: 9

```
{   { 1, 9, 8, 4 },   { 1, 8, 9, 4 },   { 2, 0, 1, 7 },   { 3, 4, 5, 8 }    }
    ^                 ^       ^
    |                 |       |
    .---.             |       |
        |             |       |
darray--.             |       |
                      |       |
p == &(darray[1])-----.       |
                              |
p[2] == &(darray[1]) + 2------.
```

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

```
char * string[];
```

A string in C is already an array of chars. So, it ends up being a char** because it is an array of an array of chars.

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

```
$ ./cmd go navy

          .---.
```

```
argv -->  | .-+--> "./cmd"
          |---|
          | . |
            .
            .
            .
```

```
            .---.
argv -->  | .-+--> "./cmd"
          |---|
          | .-+--> "go"
          |---|
          | .-+--> "navy"
          |---|
            .
            .
```

9. (10 points) Explain why the following for loop iterates over the `argv` array. (Yes, you should run this program if it helps your understand!)

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

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

}
```

A char ** is essentially an array of strings, so the loop goes until the pointer points to NULL for the arguments passed in (stored in a new copy). It then prints with the %s so that printf knows to prints until the char ** curarg is at a " ".

10. (10 points) Complete the program below that checks if each of the command line arguments is a number using `sscanf()`:

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


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

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

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

```
int trash, res;
res = sscanf(*curarg, "\%d", &trash);

if (res == 1)
```