HW 4

Joshua Thompson

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? */);
```

```
#include <stdio.h>

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;

  printf("p: (%d, %d)\n", p.left, p.right);
  printf("q: (%d, %d)\n", q->left, q->right);
  return 0;
}
```

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

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

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

```
char* s2[] = {"Beat", "Army!"};
```

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

}
```

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

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

  for(i = 0; i < strlen(s1); i++)
  {
    s2[i] = s1[i];
    printf("%c", s2[i]);
  }
  printf("\n");

}
```

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()`

   The strcpy function simply copies one string to another including the null character at the end. The first argument is the destination and the second is the one that will be copied. The destination array must be large enough to accept the string.

   (b) `strncpy()`

   Pretty much the same thing as the strcpy except it also takes an argument which chooses how many bytes you want to be copied.

   (c) `strcat()`

   Appends the input string to the destination string. Removes the null terminator at the end of the destination string, adds the other string then re-adds the null terminator at the end.

   (d) `strfry()`

   Randomizes the contents of the input string.

   (e) `strchr()`

   The function takes a string argument and a character argument. Returns a pointer to the first occurrence of the character argument.

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]);
}
```

The code given is incorrect and will not compile if simply copied an pasted.
The line int* p = (darray[1]);
Should be int* p = daray[1]; The originall was assigning int** to p which is int*.
The output is 9.

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

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

This is a double array because it is an array of Strings. Technically Strings are arrays of characters so therefore this is a double array of characters.

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

```
$ ./cmd go navy

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

verbatim .—. argv[0]-¿ — .-+–¿ ”./cmd” argv[1]-¿ — .-+–¿ ”go” argv[2]-¿ — .-+–¿ ”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);
  }

}
```

The loop prints out all of what is inside argv because a double array of characters is created in curarg. In the first part of the for loop the address of curarg is pointing to argv. The for loop continues while *curarg still has things to point to. Then at the end it increments curargy to point to the next string in argv.

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 main( int argc, char *argv[]){
char ** curarg;
int i=0;
printf("argc = %d\n", argc);
 for( curarg=argv; *curarg ; curarg++){
   int n;
   if(sscanf(argv[i], "%d", &n) == 0)
     printf("argv[%d] = %s (is *NOT* a number!)\n", i++, *curarg);
   else
     printf("argv[%d] = %s (is a number!)\n", i++, *curarg);
 }
}
```