CSE 220 – C Programming

Command Line Arguments

Outline

- Input and Output
- Command Line Arguments
- String Functions

Scanf and Printf

- Scanf and printf read and write to standard in and standard out.
- When you invoke (run) your code from the command line
  - Standard in (stdin) is the keyboard
  - Standard out (stdout) is the screen
- On HackerRank
  - stdin is a TestCase (or your custom input)
  - stdin is a message displayed to you in the output box
- However, you can also use files for stdin and stdout.

Redirection

- Scanf, printf, getchar, putchar, gets, puts: obtain input from standard input and show output to standard output (*by default*).
- Redirect input using <
- Redirect output using >

```
./factorial >output.dat

./quadratic <values.dat

./demo <in.txt >out.txt
```

Command Line Arguments

- Programs are useful because they can respond differently depending on the input they receive.
- The way you give a program input is to supply different input to stdin.
- However, there is another mechanism to pass information to programs: Command Line Arguments
- You've seen them before with the program "gcc":

```
gcc io.c -o io
```

- `"io.c"`, `"-o"`, and `"io"` are three command line arguments passed as strings to the main function.

argc and argv

- We've been defining our main function as taking no arguments (void), but most main functions actually take 2 arguments, argc and argv.
- argc is an integer representing the number of command line arguments provided to the program.
- argv is a array of strings, with each string being a command line argument.
- Example:

```
./test_program 57 -d josh
```

argc is `4`

argv is `{"./test_program", "57", "-d", "josh"}`

Using argc and argv

```
#include <stdio.h>
int main(int argc, char * argv[]) {
	printf("argc is %d\n", argc);
	for (int i = 0; i < argc; ++i) {
		printf("The arg in position %d is"
		" %s\n", i, argv[i]);
	}
}
```

```
//Contents of index.c
#include <stdio.h>
int main(int argc, char * argv[]) {
        printf("%s", argv[2]);
        return 0;
}
```

What is the output of:

```
./index josh 7 -6
```

1. ./index
2. josh
3. 7
4. -6

```
//Contents of index.c
#include <stdio.h>
int main(int argc, char * argv[]) {
        printf("%s", argv[2]);
        return 0;
}
```

What is the output of:

```
./index josh <input.txt 7 -6
```

1. ./index
2. josh
3. 7
4. -6

Converting to and from strings

- You already know the scanf and printf functions that take a format string and a series of variables to read in from stdin or write out to stdout.
- There are two functions (named "sscanf" and "sprintf") that read in from a string and write out to a string.

Examples:

- sscanf Usage:

```
char string_input[] = "56 J";

int num;

char letter;

sscanf(string_input, "%d %c", &num, &letter);
```

- sprintf Usage:

```
char string_output[100];

sprintf(string_output, "%d %c", num, letter);
```

```
//Contents of index2.c
#include <stdio.h>
int main(int argc, char * argv[]) {
        int x;
        sscanf(argv[1], "%d", &x);
        printf("%d", x * 2);
        return 0;
}
```

What is the output of:

```
./index2 4
```

1. 4
2. x
3. 8
4. Error

Practice Problems

```
//Contents of index3.c
#include <stdio.h>
int main(int argc, char * argv[]) {
	int x; char ch;
	sscanf(argv[1], "%d", &x);
	sscanf(argv[2], "%c", &ch);
	for (int i = 0; i < x; ++i) {
		printf("%c", ch);
	}
	return 0;
}
```

What is the output of:

```
./index3 3 i
```

1. 3 i
2. iii
3. ./index3
4. Error