Questions about the previous lab

Questions about HW1 (or HW0)
HW1 submit instructions

Recap from Lab 1
- Intro to Unix, Hardware, Server-Client relationships, concept behind telnet
- Intro to C
- Basic structure of a program
- Compiling and running programs
- Variables, and assigning values to them
- Data types and I/O

Recap from Lab 2
- Details on printf
- Details on scanf
- Conversion between data types
- Math operators
- Command Line Parameters
Math ops continued

• +, -, *, /, %
• ++, --
• +=, -=, *=, /=

Other symbols

• <, >, <<, >>,
• !, !=
• & & & ||
• #
• 0, 0, []

Arrays

• What are arrays?
  - Arrays are sets of consecutive memory locations used to store data
• Typical array declaration
  - int data_list[3];
  - data_list[0], data_list[1], data_list[2]
• Dimensionality
• What is the index?
• You can also initialize by doing the following
  • int data_list[3] = {1.0, 2.0, 3.0};
```c
#include <stdio.h>

#define N 5

int main (void) {
    float a[N], total, average;
    a[0] = 34.0;
    a[1] = 27.0;
    a[2] = 45.0;
    a[3] = 82.0;
    a[4] = 22.0;
    average = total/5.0;
    printf("Total is %f and Average is %f\n", total, average);
    return(0);
}
```

Multidimensional arrays

- int matrix [2][3];
- Now you assign and reference by saying
  - matrix [0][0];
  - matrix [0][1];
  - matrix [0][2];
  - matrix [1][0];
  - matrix [1][1];
  - matrix [1][2];

Strings

- Sequence of chars (an array of characters)
  #include <stdio.h>

```c
int main (void) {
    char name[6];
    name = "Suhit";
    printf("My name is %s\n", name);
    return(0);
}
```
Strings

- Sequence of chars (an array of characters)
  ```c
#include <stdio.h>

int main (void) {
    char name[6];
    name = "Suhit"; // This is wrong
    printf("My name is %s\n", name);
    return(0);
}
```

Strings II

```c
#include <stdio.h>

int main (void) {
    char name[6];
    name[0] = 'S';
    name[1] = 'u';
    name[2] = 'h';
    name[3] = 'i';
    name[4] = 't';
    name[5] = '\0'; // adding a null character at the end of the string
    printf("My name is %s\n", name);
    return(0);
}
```

Strings III

- `#include <string.h>`
  - to include special string manipulation thingies
    - `strcpy`
    - `strcmp`
    - `strlen`
    - `strcat`
    - `strtok`
Strings IV

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

int main (void) {
    char name[6];
    // One character at the end is stored for null
    strcpy(name, "Suhit");
    printf("My name is %s\n", name);
    return(0);
}
```

Strings V

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

int main (void) {
    char name[60];
    // Last character is still reserved for null, store at most 59 characters
    strcpy(name, "Suhit");
    printf("My name is %s\n", name);
    return(0);
}
```

Strings VI

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

char first_name [100];
char last_name [100];
char full_name [200];

int main (void) {
    strcpy(first_name, "Suhit");
    strcpy(last_name, "Gupta");
    strcpy(full_name, "Suhit Gupta");
    printf("My full name is %s\n", full_name);
    return(0);
}
```
Strings VII – Reading Strings

- `fgets(name, sizeof(name), stdin);`
  - `name` is the name of the character array
  - `sizeof` tells the program how much to read
  - `stdin` – keyboard

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

char line[100];

int main () {
    printf("Enter a line: ");
    fgets(line, sizeof(line), stdin);
    printf("The length of the line is \%d\n", strlen(line));
    return(0);
}
```

Strings VIII

- `fgets` has last character as end-of-line (newline)
- Some people will munge the last newline char by doing the following
  - `line[strlen(line)-1] = \'\0\`
- Then use `sscanf` – like `scanf`, but used to scan strings
  - Usage : `sscanf(in_string, "%d%d%d%s", &a, &b, &c, tmp);`

BTW…

- In Ch. 5, read about different data types, like different types of int, types of float.
- Also read about hexadecimal and octal
- We will cover this in depth as the course goes on
Loops and conditionals

- **if**
  - need to know `<`, `>`, `==`, `!=`
  - usage: `if (expr) {stmt…}
    else if (expr) {stmt…}
    else {stmt}

- **while**
  - usage: `while (cond) {stmt…}
  - break;

Next time…

- Iteration/loops
  - While
  - For
  - Do while
- Conditional statements
  - If
  - Switch
- Methods and method calls
  - Variable scope
  - Return values

Assignment

- Read Ch. 6 from the Practical C Programming book
- HW1