Module 3 Lab:

Problem 1:

Read the strings entered by the user and print the largest line among them.

Source Code:

```c
// UCI DCE 22 Fall
//
// EECS_805: C Programming for Embedded System
// Module 3 Lab
// Problem 1
//
// Cheng Fei

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

// Macro variables settings.
#define LF 10
#define CR 13
#define MAX_STRING_NUMBER 100
#define MAX_CHARACTER_NUMBER 100

/*************************/
/* Main Function         */
/*************************/
// Prompt the user to enter strings and output the largest string the user enters.
int main(void) {
  // Basic variables settings.
  char strings[MAX_STRING_NUMBER][MAX_CHARACTER_NUMBER] = {0}; // Default maximum length of a single string is 100 characters.
  int stringIndex = 0; // Record the index of the accumulative longest string.
  int longestLength = 0; // Record the accumulative largest length of strings.
  int stringi = 0; // Index of the current string.

  // Ouput the prompt.
  printf("Please input the strings: \n");
  printf("*************************************************\n");
  printf("*** Separate sentences with the return keystroke ***\n");
  printf("*** End by entering X                            ***\n");
  printf("*************************************************\n");

  // Read from the input device.
  do {
    // Read strings separated by whitespace/newline/tab.
    scanf("%s", strings[stringi]);
    stringi++;
  } while (strings[stringi-1][0] != 'X'); // Terminate when entering 'X'.

  // Retrieve the longest string.
  for (int i = 0; i <= stringi; i++) {
    if (strlen(strings[i]) > longestLength) {
      longestLength = strlen(strings[i]);
      stringIndex = i;
    }
  }

  // Output the longest string.
  printf("The longest string you entered is: \n");
  printf("%s\n", strings[stringIndex]);

  return EXIT_SUCCESS;
}
```

Console Output:

```
Please input the strings:
**************************************************
*** Separate sentences with the return keystroke ***
*** End by entering X                            ***
**************************************************
This is a great assignment, I learned the standard I/O library and understood the difference between I/O methods, like scanf, getc, getch, getchar!
X
The longest string you entered is:
assignment,
```

Problem 2:

2.1.

Prompt:

What is the output for the following code?

printf("Hello World" + 4);

Answer:

Warning: adding 'int' to a string does not append to the string

Source Code:

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

int main(void) {
    printf("Hello World" + 4);
    return EXIT_SUCCESS;
}
```

Console Output:

```
main.c:12:24: warning: adding 'int' to a string does not append to the string [-Wstring-plus-int]
  printf("Hello World" + 4);
         ~~~~~~~~~~~~~~^~~
main.c:12:24: note: use array indexing to silence this warning
  printf("Hello World" + 4);
                       ^
         &             [  ]
1 warning generated.
```

2.2.

Prompt:

What would be the output of the following C program?

Answer:

4321

Source Code:

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

int main(void) {
  int i = 43;
  printf("%d",printf("%d",printf("%d",i)));
  return EXIT_SUCCESS;
}
```

Console Output:

```
4321
```

Problem 3:

Prompt:

Why must x be preceded by & inside scanf?

Answer:

Passing x as a reference enables the function scanf to actually change the value of x outside of the scanf's scope. If otherwise, passing x as a copy (not preceded by &), the scanf function will simply make a shallow copy of the object and only stores the input value in this copy. However, as a local variable to this function, this copy of variable x will be deleted (the designated memory will be released) after the function stack is closed; that being said, the input value of the scanf function will be discarded.

Problem 4:

Prompt:

Why getchar return value is of type int?

Answer:

The getchar function returns the character it reads, and based on the ASCII/UTF, every character has a unique index code. I think returning an integer can better represent some special characters, like the null terminator (‘\0’), the newline character (‘\n’), etc., since these characters are normally invisible in the text.