Assignment 1:

Solution 1:

```
#include <stdio.h>  //including the standard input and the output header file
int main(){
   printf("Hello\n"); //String constant since it enclosed in the double quotes
   printf("Student");  //String constant
   return 0;
}
```

Solution 2:

```
#include <stdio.h>  //including the standard input and the output header file
int main(){
   printf("Hello\n"); //String constant since it enclosed in the double quotes
   printf("Student");  //String constant
   return 0;
}
```

Solution 4:

```
#include <stdio.h>  //including the standard input and the output the output header fi
int main(){
   int r;  //declaring the radius variable
   double A;
   printf("Enter the radius of the circle:\n");
   scanf("%d",&r);  //taking the input from the user
   A = 3.14*r*r;
   printf("Area of circle is %f having the radius %d",A,r);
   return 0;   //Telling the operating system that the program executed successfully
}
```

Solution 5:

```
#include <stdio.h> //including the standard input and output header file
int main(){     //declaring the main function
   char a[100]; //declaring the character array
```

```
   printf("Enter the sentence whose length is to be calculated\n");
   int length;   //declaring the integer variable
   gets(a);      //taking the input using the gets function
   length = strlen(a);
   printf("The length of the string is %d",length);
   return 0;  //telling the os that the the program is exited successfully
}
```

Solution 6:

```
#include <stdio.h>  //including the standard input and the output header header file
int main(){  //declaring the main function
   printf("\"Hello, Amit Kumar\""); //using the escape the sequence to print the double quotes
   return 0; //Telling that the os has successfully executed the program
}
```

Solution 7:

```
#include <stdio.h> //including the standard input and the output header file
int main(){  //declaring the main function
   printf("\"\%%d\"");  //using the escape sequence for printing the special characters " and %d
   return 0; //Telling the os that the program has successfully executed
}
```

Solution 8:

```
#include <stdio.h> //including the standard input and output header file
int main(){
   printf("\"\\n\""); //using the escape sequence for the special characters " and \n
   return 0;    //Telling the os that the program has successfully executed
}
```

Solution 9:

```
#include <stdio.h>  //including the standard input and the output header file
int main(){   //declaring the main function
   printf("\"\\\\\"");  //for printing the / we use // and for printing // we use ////
   return 0; //Telling the os that the program has successfully executed
}
```

Solution 10:

```
#include <stdio.h>

// Function to convert date format
void convert_date_format(char *input_date) {
   int day, month, year;
   sscanf(input_date, "%d/%d/%d", &day, &month, &year);
   printf("Day - %02d , Month - %02d , Year - %d\n", day, month, year);
}

int main() {
   char user_input[11]; // DD/MM/YYYY + null terminator

   // Get user input
   printf("Enter a date in DD/MM/YYYY format: ");
   scanf("%s", user_input);

   // Convert and display the result
   convert_date_format(user_input);

   return 0;
}
```

This explains that why %02d is used in place of the format specifier %d.

The format specifier `%02d` is used to format an integer value with leading zeros so that it always takes up at least two characters of width, and if the value is less than 10, it will be padded with a leading zero. Let's break down the format specifier and its purpose:

- `%` : Denotes the beginning of a format specifier.

- `0` : Indicates that if the field width is less than the specified width, it should be padded with zeros.

- `2` : Specifies the minimum field width to be used. It ensures that the printed value occupies at least two characters.

- `d` : Specifies that the argument to be printed is an integer.

For example, consider the following values:

- If the integer is `5`, using `%02d` would result in `"05"`.
- If the integer is `12`, using `%02d` would result in `"12"` (no padding needed).

In the context of formatting dates, the use of `%02d` ensures that single-digit day and month values are padded with leading zeros to maintain a consistent width, making the output more visually appealing. This is particularly useful for aligning data in tabular formats or when dealing with data that needs to be consistently formatted.

If you were to use `%d` instead of `%02d`, single-digit values would not be padded with leading zeros, potentially affecting the alignment and visual presentation of the output.

This line explains the use of the sscanf in place of the scanf in the above program:

In the C code provided, the `sscanf` function is used to extract data from a string based on a specified format, whereas the `scanf` function is used to read data from standard input (usually the keyboard) based on a specified format. Let's explore the differences between these two functions:

1. **`sscanf` Function:**
   - The `sscanf` function is used for **reading from a string** instead of directly from standard input.
   - It takes three main arguments:
     - The input string containing the data you want to extract from (`input_date` in this case).
     - A format string that specifies the expected format of the input data.
     - A variable list of pointers to the variables where the extracted values should be stored.
   - `sscanf` uses the format specifier `%d` to match and extract integer values.

2. **`scanf` Function:**
   - The `scanf` function is used for **reading from standard input** (keyboard, typically).
   - It also takes a format string as an argument, but it reads data directly from the keyboard.
   - The format specifiers used in `scanf` are the same as those used in `printf`.
   - For reading integers, `%d` is used as the format specifier.

In the provided C code, `sscanf` is used because the date input is already stored in the `user_input` string. The goal is to extract the day, month, and year components from this string using the format specifier `%d/%d/%d`.

Using `scanf` in this context would require reading directly from the keyboard, which isn't suitable since you want to process the date stored in the `user_input` string.

To summarize, the `sscanf` function is used here to parse the date string according to a specific format, while `scanf` is not suitable because it reads data from the standard input stream rather than from a string.

Solution 11:

```
#include <stdio.h>  //including the standard input and the output header file
void specified_format(char *output_format){  //creating the specified format function
   int hour, minute;                    //declaring the hour and the minute variable as integer
   sscanf(output_format,"%d:%d",&hour,&minute);
   printf("%02d hour and %02d minute",hour,minute);
}

int main(){
   char user_input[6];   //creating the character array of length 6 including the null character
   printf("Enter the time in HH:MM format: ");
   scanf("%s",user_input);
   specified_format(user_input);   //passing the array to the specified format function
   return 0;     //Telling the os that the function has successfully terminated
}
```

Solution 12:

```
#include <stdio.h>  //including the standard input and the output header file
int main(){        //creating the main function
   int x = printf("ineuron");  //here the first print statement prints ineuron and return the number of characters which is 7 and the second print statement gives the value 7
```

```
    printf("%d",x);
    return 0;      //Telling the os that the function has successfully terminated
}
```