



San Francisco Bay University

CS360L - Programming in C and C++ Lab Lab Assignment #1

Due day: 5/24/2022

Instruction:

1. Push the answer sheets/source code to Github
2. Please follow the code style rule like programs on handout.
3. Overdue lab assignment submission can't be accepted.
4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)

1. Let's examine / run the following C++ program regarding *string* data type and related operators

```
#include <iostream>
```

```
using std::cout;  
using std::endl;  
using std::string;
```

```
const char SEMI_COLON = ':';  
const string VERB1 = "went up ";  
const string VERB2 = "down came ";  
const string VERB3 = "washed ";  
const string VERB4 = "out came ";  
const string VERB5 = "dried up ";
```

```
int main(void){  
    string firstLine;  
    string secondLine;  
    string thirdLine;  
    string fourthLine;
```

```
    firstLine = "The itsy bitsy spider " + VERB1 + "the water spout";  
    secondLine = VERB2 + "the rain and " + VERB3 + "the spider out";  
    thirdLine = VERB4 + "the sun and " + VERB5 + "all the rain";  
    fourthLine = "and the itsy bitsy spider " + VERB1 + "the spout again";
```

```
    cout << firstLine << SEMI_COLON << endl;  
    cout << secondLine << SEMI_COLON << endl;  
    cout << thirdLine << SEMI_COLON;  
    cout << endl;  
    cout << fourthLine << '.' << endl;
```

```

    return 0;
}

```

```

01.cpp
lab1_1 > C- 01.cpp
12  const string VERB5 = "dried up ";
13
14  int main(void)
15  {
16      string firstLine;
17      string secondLine;
18      string thirdLine;
19      string fourthLine;
20
21      firstLine = "The itsy bitsy spider " + VERB1 + "the water spout";
22      secondLine = VERB2 + "the rain and " + VERB3 + "the spider out";
23      thirdLine = VERB4 + "the sun and " + VERB5 + "all the rain";
24      fourthLine = "and the itsy bitsy spider " + VERB1 + "the spout
again";
25
26      cout << firstLine << SEMI_COLON << endl;
27      cout << secondLine << SEMI_COLON << endl;
28      cout << thirdLine << SEMI_COLON;
29      cout << endl;
30      cout << fourthLine << '.' << endl;
31      return 0;
32  }
33
~/CS360/lab1_1$ ./1.Output
The itsy bitsy spider went up the water spout;
down came the rain and washed the spider out;
out came the sun and dried up all the rain;
and the itsy bitsy spider went up the spout again.
~/CS360/lab1_1$

```

2. Focuses on constructing output statements. Program Shell is the outline of a program. Use this shell for Question#1 through #3
 - a. Question#1: Write a program to read-in from keyboard and print the following information single spaced on the screen. Use literal constants in the output statements for each of the data items to be written on the screen. Run your program to verify that the output is as specified.
 - i. your name (last name, comma, blank, first name)
 - ii. today's date (month:day:year)

CODE:

```

#include <iostream>
#include <string>

```

```

using namespace std;

```

```

int main() {
    string firstName, lastName;
    int month, day, year;

```

```

    cout << "Enter your first name: ";
    cin >> firstName;
    cout << "Enter your last name: ";
    cin >> lastName;

```

```

    cout << "Enter today's date (month day year, separated by spaces): " << endl;
    cin >> month >> day >> year;

```

```

    cout<< "The output will be shown here : "<<endl;
    cout << lastName << ", " << firstName << endl;
    cout << (month < 10 ? "0" : "") << month << ":"
        << (day < 10 ? "0" : "") << day << ":" << year << endl;

    return 0;
}

```

OUTPUT:



```

4  using namespace std;
5
6  int main() {
7      string firstName, lastName;
8      int month, day, year;
9
10     cout << "Enter your first name: ";
11     cin >> firstName;
12     cout << "Enter your last name: ";
13     cin >> lastName;
14
15     cout << "Enter today's date (month day year, separated by spaces): "
16     <<endl;
17     cin >> month >> day >> year;
18
19     cout<< "The output will be shown here : "<<endl;
20     cout << lastName << ", " << firstName << endl;
21     cout << (month < 10 ? "0" : "") << month << ":"
22         << (day < 10 ? "0" : "") << day << ":" << year << endl;
23
24     return 0;
25 }

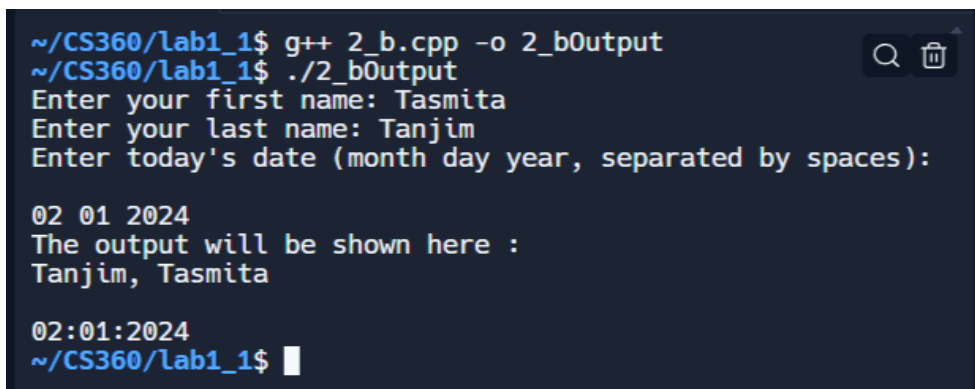
```

```

~/CS360/Lab1_1$ g++ 2_a.cpp -o 2_aOutput
~/CS360/Lab1_1$ ./2_aOutput
Enter your first name: Tasmita
Enter your last name: Tanjim
Enter today's date (month day year, separated by spaces):
02 01 2024
The output will be shown here :
Tanjim, Tasmita
02:01:2024
~/CS360/Lab1_1$

```

- b. Question#2: Change your program so that there is a space between the two output lines.



```

~/CS360/Lab1_1$ g++ 2_b.cpp -o 2_bOutput
~/CS360/Lab1_1$ ./2_bOutput
Enter your first name: Tasmita
Enter your last name: Tanjim
Enter today's date (month day year, separated by spaces):

02 01 2024
The output will be shown here :
Tanjim, Tasmita

02:01:2024
~/CS360/Lab1_1$

```

- c. Question#3: Change your program so that your first name is printed followed by your last name, with a blank in between the names.

```

int main() {
    string firstName, lastName;
    int month, day, year;

    cout << "Enter your first name: ";
    cin >> firstName;
    cout << "Enter your last name: ";
    cin >> lastName;

    cout << "Enter today's date (month day year, separated by spaces):";
    cin >> month >> day >> year;

    cout<< "The output will be shown here: "<<endl;
    cout << firstName << ", " << lastName << endl;
    cout<<endl;
    cout << (month < 10 ? "0" : "") << month << ":"
        << (day < 10 ? "0" : "") << day << ":" << year << endl;

    return 0;
}

```

```

~/CS360/lab1_1$ g++ 2_c.cpp -o 2_cOutput
~/CS360/lab1_1$ ./2_cOutput
Enter your first name: Tasmita
Enter your last name: Tanjim
Enter today's date (month day year, separated by spaces):
02 01 2024
The output will be shown here:
Tasmita, Tanjim

02:01:2024
~/CS360/lab1_1$

```

3. Use the following program shell for Question#1 through #3

- a. Question#1: Write a named string constant made up of your first and last names with a blank in between. Write the statements to print out the result of applying *length* and *size* to your named constant object. Compile and run your program.

CODE:

```

#include <iostream>
#include <string>

using namespace std;

int main(void) {

```

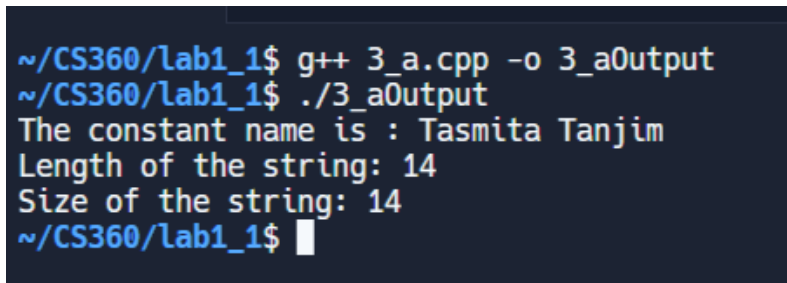
```

const string name = "Tasmita Tanjim";

cout<<"The constant name is : "<<name<<endl;
cout << "Length of the string: " << name.length() << "\n";
cout << "Size of the string: " << name.size() << "\n";

return 0;
}

```



```

~/CS360/lab1_1$ g++ 3_a.cpp -o 3_aOutput
~/CS360/lab1_1$ ./3_aOutput
The constant name is : Tasmita Tanjim
Length of the string: 14
Size of the string: 14
~/CS360/lab1_1$ 

```

- b. Question#2: Add statements to your Question#1 program to print your name formatted as last name first, followed by a comma and your first name. Use function *substr* to accomplish this task. Compile and run your program.

CODE:

```

#include <iostream>
#include <string>

using namespace std;

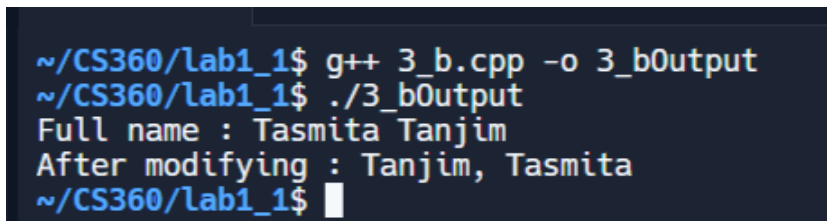
int main() {
    const string fullName = "Tasmita Tanjim";
    cout<<"Full name : "<<fullName<<endl;

    const size_t spaceIdx = fullName.find(' ');
    const string lastName = fullName.substr(spaceIdx + 1);
    const string firstName = fullName.substr(0, spaceIdx);

    cout << "After modifying : "<<lastName << ", " << firstName << endl;

    return 0;
}

```



```

~/CS360/lab1_1$ g++ 3_b.cpp -o 3_bOutput
~/CS360/lab1_1$ ./3_bOutput
Full name : Tasmita Tanjim
After modifying : Tanjim, Tasmita
~/CS360/lab1_1$ 

```

- c. Question#3: Add the statements necessary to print your last name, followed by a comma and your first initial. Compile and run your program

CODE:

```
#include <iostream>
#include <string>

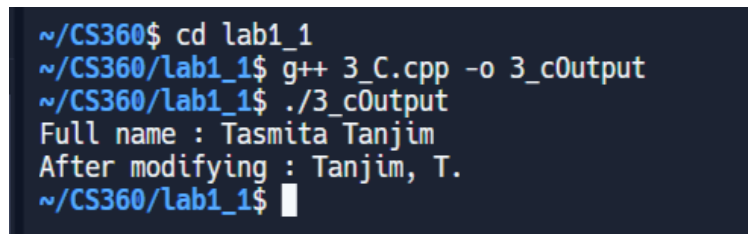
using namespace std;

int main() {
    const string fullName = "Tasmita Tanjim";
    cout<<"Full name : "<<fullName<<endl;

    const size_t spaceIndex = fullName.find(' ');
    const string lastName = fullName.substr(spaceIndex + 1);
    const char firstInitial = fullName[0];

    cout << "After modifying : "<<lastName << ", " << firstInitial << '.' << endl;

    return 0;
}
```



```
~/CS360$ cd lab1_1
~/CS360/lab1_1$ g++ 3_C.cpp -o 3_cOutput
~/CS360/lab1_1$ ./3_cOutput
Full name : Tasmita Tanjim
After modifying : Tanjim, T.
~/CS360/lab1_1$
```

4. Use the following program shell for Question#1 through Question#4

- a. **Question#1:** Write a program to print the following numbers **right-justified** in a column on the screen. Make the values named constants.

1066 1492 512 1 -23

ANSWER:

```
#include <iostream>
#include <iomanip>
#include <algorithm>

using namespace std;

const int NUMBER1 = 1066;
```

```

const int NUMBER2 = 1492;
const int NUMBER3 = 512;
const int NUMBER4 = 1;
const int NUMBER5 = -23;

int main() {
    cout << fixed << showpoint;

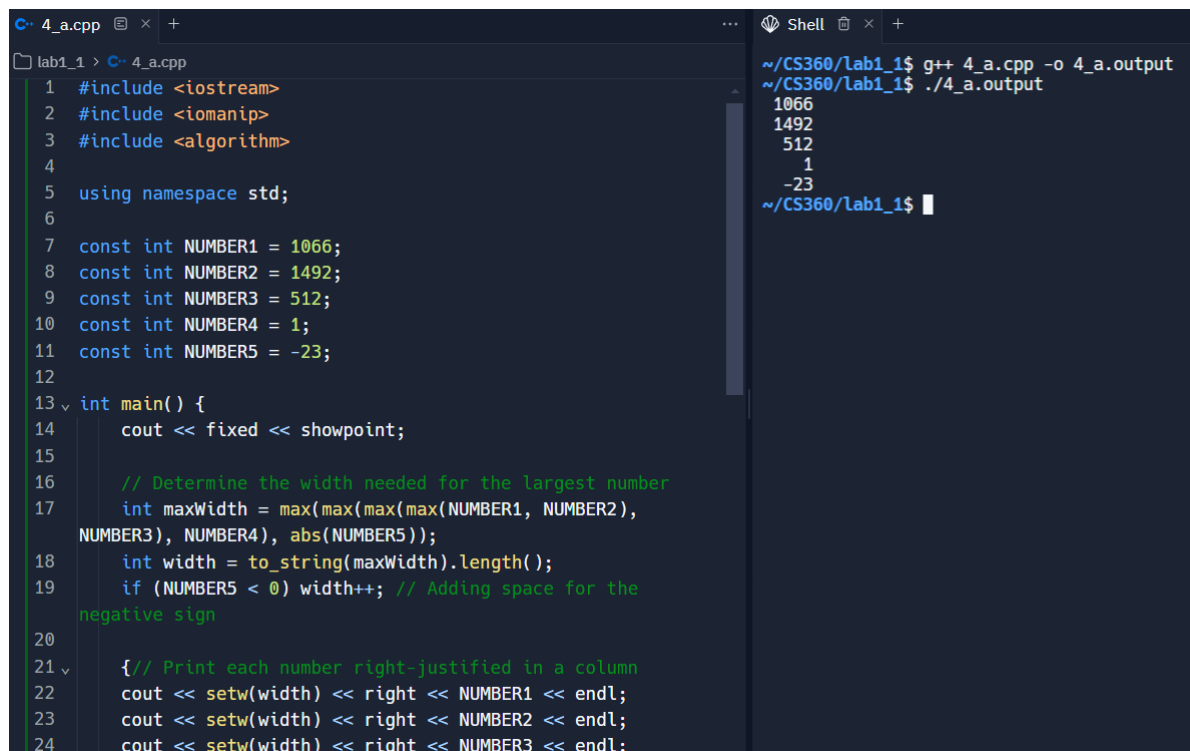
    // Determine the width needed for the largest number
    int maxWidth = max(max(max(max(NUMBER1, NUMBER2), NUMBER3),
NUMBER4), abs(NUMBER5));
    int width = to_string(maxWidth).length();
    if (NUMBER5 < 0) width++; // Adding space for the negative sign

    { // Print each number right-justified in a column
        cout << setw(width) << right << NUMBER1 << endl;
        cout << setw(width) << right << NUMBER2 << endl;
        cout << setw(width) << right << NUMBER3 << endl;
        cout << setw(width) << right << NUMBER4 << endl;
        cout << setw(width) << right << NUMBER5 << endl;
    }

    return 0;
}

```

OUTPUT:



The screenshot shows a code editor with the source code of the program and a terminal window displaying the output. The source code defines five constants: NUMBER1 (1066), NUMBER2 (1492), NUMBER3 (512), NUMBER4 (1), and NUMBER5 (-23). The main function uses `cout` with `fixed` and `showpoint` to print each number right-justified in a column. The output in the terminal shows the numbers 1066, 1492, 512, 1, and -23, each on a new line, right-aligned.

```

1  #include <iostream>
2  #include <iomanip>
3  #include <algorithm>
4
5  using namespace std;
6
7  const int NUMBER1 = 1066;
8  const int NUMBER2 = 1492;
9  const int NUMBER3 = 512;
10 const int NUMBER4 = 1;
11 const int NUMBER5 = -23;
12
13 int main() {
14     cout << fixed << showpoint;
15
16     // Determine the width needed for the largest number
17     int maxWidth = max(max(max(max(NUMBER1, NUMBER2),
NUMBER3), NUMBER4), abs(NUMBER5));
18     int width = to_string(maxWidth).length();
19     if (NUMBER5 < 0) width++; // Adding space for the
negative sign
20
21     { // Print each number right-justified in a column
22         cout << setw(width) << right << NUMBER1 << endl;
23         cout << setw(width) << right << NUMBER2 << endl;
24         cout << setw(width) << right << NUMBER3 << endl;

```

```

~/CS360/lab1_1$ g++ 4_a.cpp -o 4_a.output
~/CS360/lab1_1$ ./4_a.output
1066
1492
512
1
-23
~/CS360/lab1_1$

```

- b. Question#2: Add two statements to your program. Calculate the floating-point result from dividing the sum of the first two values by the sum of the last three values and store it in answer. The second statement should write the contents of answer on the screen to four decimal places. (Do not forget to declare *answer*.)

The answer is _____.

CODE:

```
#include <iostream>
#include <iomanip>

using namespace std;

const int NUMBER1 = 1066;
const int NUMBER2 = 1492;
const int NUMBER3 = 512;
const int NUMBER4 = 1;
const int NUMBER5 = -23;

int main() {
    // Calculate the sums and the floating-point result
    double sumFirstTwo = NUMBER1 + NUMBER2;
    double sumLastThree = NUMBER3 + NUMBER4 + NUMBER5;
    double answer = sumFirstTwo / sumLastThree; // Ensure floating-point division

    // Set a fixed width for displaying the numbers
    int width = 6; // Chosen to accommodate the largest number and potential negative sign

    // Print each number right-justified in a column
    cout << setw(width) << right << NUMBER1 << endl;
    cout << setw(width) << right << NUMBER2 << endl;
    cout << setw(width) << right << NUMBER3 << endl;
    cout << setw(width) << right << NUMBER4 << endl;
    cout << setw(width) << right << NUMBER5 << endl;

    // Set precision for the answer
    cout << "The answer is " << setprecision(4) << fixed << answer << endl;

    return 0;
}
```



```
4_b.cpp 4_b.cpp
lab1_1 > 4_b.cpp

11
12 int main() {
13     // Calculate the sums and the floating-point result
14     double sumFirstTwo = NUMBER1 + NUMBER2;
15     double sumLastThree = NUMBER3 + NUMBER4 + NUMBER5;
16     double answer = sumFirstTwo / sumLastThree; // Ensure
floating-point division
17
18     // Set a fixed width for displaying the numbers
19     int width = 6; // Chosen to accommodate the largest
number and potential negative sign
20
21     // Print each number right-justified in a column
22     cout << setw(width) << right << NUMBER1 << endl;
23     cout << setw(width) << right << NUMBER2 << endl;
24     cout << setw(width) << right << NUMBER3 << endl;
25     cout << setw(width) << right << NUMBER4 << endl;
26     cout << setw(width) << right << NUMBER5 << endl;
27
28     // Set precision for the answer and print it
29     cout << "The answer is " << setprecision(4) << fixed <<
answer << endl;

~/CS360/lab1_1$ g++ 4_b.cpp -o 4_bOutput
~/CS360/lab1_1$ ./4_bOutput
1066
1492
512
1
-23
The answer is 5.2204
~/CS360/lab1_1$
```

- c. Question#3: Write the following numbers **right-justified** in a column on the screen. Each of the data values should be written in formatted floating-point notation with two decimal places. Use field width specifications rather than listing the numbers in your program with the proper formatting. You may use either literal constants or named constants.

23.62 46.0 43.4443 100.1 98.98

CODE:

```
#include <iostream>
#include <iomanip>
```

```
using namespace std;
```

```
const double NUMBER1 = 23.62;
const double NUMBER2 = 46.0;
const double NUMBER3 = 43.4443;
const double NUMBER4 = 100.1;
const double NUMBER5 = 98.98;
```

```
int main() {
    cout << fixed << setprecision(2);
```

```
    int width = 6;
```

```
    cout << setw(width) << right << NUMBER1 << endl;
    cout << setw(width) << right << NUMBER2 << endl;
    cout << setw(width) << right << NUMBER3 << endl;
    cout << setw(width) << right << NUMBER4 << endl;
    cout << setw(width) << right << NUMBER5 << endl;
```

```

    return 0;
}

```

The screenshot shows a C++ IDE with a file named `4_c.cpp`. The code defines five constant double variables: `NUMBER1 = 23.62`, `NUMBER2 = 46.0`, `NUMBER3 = 43.4443`, `NUMBER4 = 100.1`, and `NUMBER5 = 98.98`. The `main` function sets the output precision to 2 and the width to 6, then prints each number using `setw` and `right` alignment. The terminal output shows the numbers formatted as 23.62, 46.00, 43.44, 100.10, and 98.98.

```

1 #include <iostream>
2 #include <iomanip>
3
4 using namespace std;
5
6 const double NUMBER1 = 23.62;
7 const double NUMBER2 = 46.0;
8 const double NUMBER3 = 43.4443;
9 const double NUMBER4 = 100.1;
10 const double NUMBER5 = 98.98;
11
12 int main() {
13     cout << fixed << setprecision(2);
14
15     int width = 6;
16
17     cout << setw(width) << right << NUMBER1 << endl;
18     cout << setw(width) << right << NUMBER2 << endl;
19     cout << setw(width) << right << NUMBER3 << endl;
20     cout << setw(width) << right << NUMBER4 << endl;
21     cout << setw(width) << right << NUMBER5 << endl;
22 }

```

```

~/CS360/Lab1_1$ g++ 4_c.cpp -o 4_cOutput
~/CS360/Lab1_1$ ./4_cOutput
23.62
46.00
43.44
100.10
98.98
~/CS360/Lab1_1$

```

- d. Question#4: Add two statements to your program for Question#3. The first statement should calculate the sum of the numbers and store the result in variable `sum`. The second statement should write `sum` on the screen, properly labeled.

The sum of the numbers is _____.

CODE:

```

#include <iostream>
#include <iomanip>

```

```

using namespace std;

```

```

const double NUMBER1 = 23.62;
const double NUMBER2 = 46.0;
const double NUMBER3 = 43.4443;
const double NUMBER4 = 100.1;
const double NUMBER5 = 98.98;

```

```

int main() {
    cout << fixed << setprecision(2);

```

```

    int width = 6;

```

```

    cout << setw(width) << right << NUMBER1 << endl;
    cout << setw(width) << right << NUMBER2 << endl;
    cout << setw(width) << right << NUMBER3 << endl;
    cout << setw(width) << right << NUMBER4 << endl;
    cout << setw(width) << right << NUMBER5 << endl;

```

```

// Calculate the sum of the numbers
double sum = NUMBER1 + NUMBER2 + NUMBER3 + NUMBER4 + NUMBER5;

// Print the sum
cout << "The sum of the numbers is " << sum << endl;

return 0;
}

```

The screenshot shows a C++ IDE with a file named 4_d.cpp. The code defines five constants: NUMBER2 = 46.0, NUMBER3 = 43.4443, NUMBER4 = 100.1, and NUMBER5 = 98.98. The main function uses cout with fixed, setprecision(2), and setw(6) to format the output. The output window shows the numbers 23.62, 46.00, 43.44, 100.10, and 98.98, followed by the sum: "The sum of the numbers is 312.14".

```

lab1_1 > C:\4_d.cpp
7  const double NUMBER2 = 46.0;
8  const double NUMBER3 = 43.4443;
9  const double NUMBER4 = 100.1;
10 const double NUMBER5 = 98.98;
11
12 int main() {
13     cout << fixed << setprecision(2);
14
15     int width = 6;
16
17     cout << setw(width) << right << NUMBER1 << endl;
18     cout << setw(width) << right << NUMBER2 << endl;
19     cout << setw(width) << right << NUMBER3 << endl;
20     cout << setw(width) << right << NUMBER4 << endl;
21     cout << setw(width) << right << NUMBER5 << endl;
22
23     // Calculate the sum of the numbers
24     double sum = NUMBER1 + NUMBER2 + NUMBER3 + NUMBER4 +

```

```

~/CS360/Lab1_1$ g++ 4_d.cpp -o 4_d0output
~/CS360/Lab1_1$ ./4_d0output
23.62
46.00
43.44
100.10
98.98
The sum of the numbers is 312.14
~/CS360/Lab1_1$

```

5. Use the following program shell for Question#1through #3.

*// Program Center sends strings to the output stream in
// specified formats.*

```

#include <iostream>
#include <iomanip>

```

```

using std::cout;

```

```

int main (void){
    return 0;
}

```

- a. Question#1: Add the statements necessary to print the following strings centered in fields of 20 characters, all on one line: "Good Morning", "Sarah", and "Sunshine!". Do not use **manipulators**. Compile and run your program; show your output.

CODE:

```

#include <iostream>
#include <string>

using namespace std;

```

```

int main() {
    string greeting = "Good Morning";
    string name = "Sarah";
    string exclamation = "Sunshine!";

    int paddingGreeting = (20 - greeting.length()) / 2;
    int paddingName = (20 - name.length()) / 2;
    int paddingExclamation = (20 - exclamation.length()) / 2;

    cout << string(paddingGreeting, ' ') << greeting << string(20 - greeting.length() -
paddingGreeting, ' ');

    cout << string(paddingName, ' ') << name << string(20 - name.length() - paddingName, '
');

    cout << string(paddingExclamation, ' ') << exclamation << string(20 -
exclamation.length() - paddingExclamation, ' ');

    return 0;
}

```

The screenshot shows a C++ IDE with two panels. The left panel displays the source code for `5_a.cpp`, which includes the `main` function. The code defines three strings: "Good Morning", "Sarah", and "Sunshine!". It calculates the padding for each string to center them in a 20-character field. The right panel shows the terminal output, which displays the three strings centered in their respective fields: "Good Morning", "Sarah", and "Sunshine!".

- b. Question#2: Repeat Question#1 using **manipulators** to help center your strings. Compile and run your program. Your output should be the same.

CODE:

```

#include <iostream>
#include <iomanip>

using namespace std;

```

```

int main() {
    // Strings to be printed
    string greeting = "Good Morning";
    string name = "Sarah";
    string exclamation = "Sunshine!";

    // Calculate total padding for each string
    int totalPaddingGreeting = 20 - greeting.length();
    int totalPaddingName = 20 - name.length();
    int totalPaddingExclamation = 20 - exclamation.length();

    // Calculate padding on the left for each string to center them
    int leftPaddingGreeting = totalPaddingGreeting / 2 + greeting.length();
    int leftPaddingName = totalPaddingName / 2 + name.length();
    int leftPaddingExclamation = totalPaddingExclamation / 2 + exclamation.length();

    // Print each string centered in a field of 20 characters
    cout << setw(leftPaddingGreeting) << greeting;
    cout << setw(leftPaddingName) << name;
    cout << setw(leftPaddingExclamation) << exclamation << endl;

    return 0;
}

```

```

~ /CS360/Lab1_1$ g++ 5_b.cpp -o 5_b0output
~ /CS360/Lab1_1$ ./5_b0output
Good Morning Sarah Sunshine!
~ /CS360/Lab1_1$

```

- c. Question#3: Change the program in Question#2 so that the three strings are printed on three separate lines with a blank line in between each string.

CODE:

```

#include <iostream>
#include <iomanip>

using namespace std;

int main()

```

```

{
    // Strings to be printed
    string greeting = "Good Morning";
    string name = "Sarah";
    string exclamation = "Sunshine!";

    // Calculate total padding for each string
    int totalPaddingGreeting = 20 - greeting.length();
    int totalPaddingName = 20 - name.length();
    int totalPaddingExclamation = 20 - exclamation.length();

    // Calculate padding on the left for each string to center them
    int leftPaddingGreeting = totalPaddingGreeting / 2 + greeting.length();
    int leftPaddingName = totalPaddingName / 2 + name.length();
    int leftPaddingExclamation = totalPaddingExclamation / 2 + exclamation.length();

    // Print each string centered in a field of 20 characters
    cout << setw(leftPaddingGreeting) << greeting;
    cout << setw(leftPaddingName) << name;
    cout << setw(leftPaddingExclamation) << exclamation << endl;

    return 0;
}

```

The screenshot shows a C++ IDE with a file named `5_c.cpp` open. The code is the same as the one in the previous block. The output window on the right shows the result of running the program: `Good Morning Sarah Sunshine!`. The output is centered in a field of 20 characters for each string. The IDE also shows a terminal window with the command `g++ 5_c.cpp -o 5_cOutput` and the execution command `./5_cOutput`.

```

~/CS360/Lab1_1$ g++ 5_c.cpp -o 5_cOutput
~/CS360/Lab1_1$ ./5_cOutput
Good Morning Sarah Sunshine!
~/CS360/Lab1_1$

```