Activity No. 2.1	
Hands-on Activity 2.1: Data Types and Arithmetic Operations	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** 8/7/2025
Section: CPE11S1	**Date Submitted:** 8/7/2025
Name(s) Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**

Output

Example 1: The following program has an output of:

The value of seven is: 7.000000

The value of eight and a half is: 8.500000

Can you find all possible compilation errors and logic errors? Can you fix them to print the same result as the expected output? Before you use your compiler, try to find the errors only by manual code analysis.

CODE (with corrections):

```
#include<iostream>
put #include <iomanip> to be able to use for std::setprecision to be able to display 7.000000 and 8.500000 or else it will just display 7 and 8.5 only

using namespace std;

int main()

{

// add cout << fixed << std::setprecision(6); to display 7.000000 and 8.500000

cout<<"The value of seven is: "; // put add <<7.000000<< std::endl; to display “7.000000” and to make space for the next line of code.

cout<<"The value of eight and a half is: ", <<8.5; // remove the “,” and type 8.500000

return 0;

}
```

CODE (corrections applied):

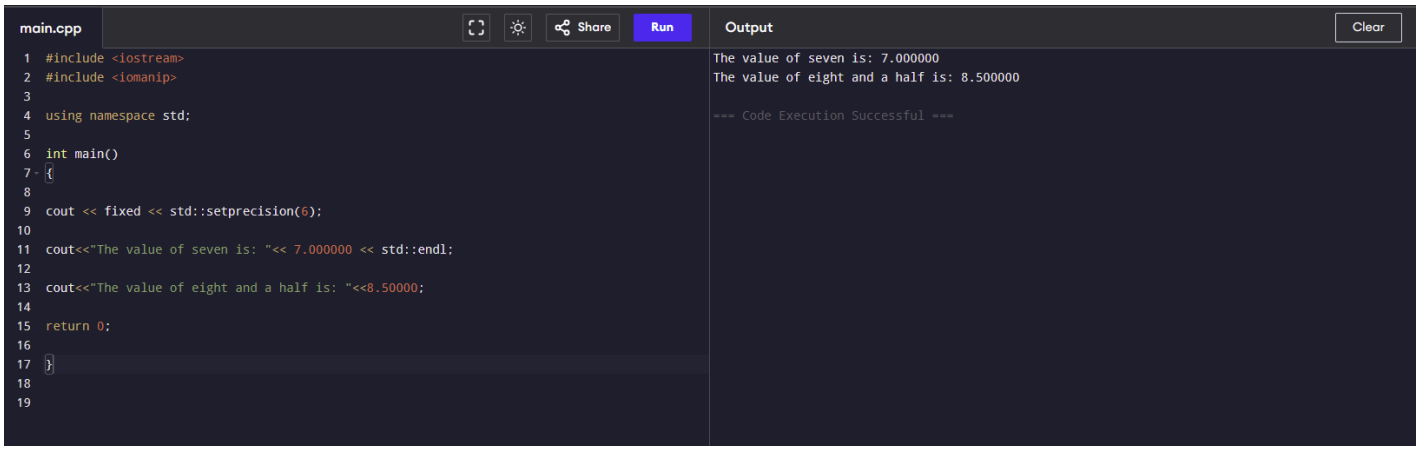
```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{

cout << fixed << std::setprecision(6);

cout<<"The value of seven is: "<< 7.000000 << std::endl;
```

cout<<"The value of eight and a half is: "<<8.50000;

return 0;

}

RESULT:



Example 2: The following program has an output of:

The value of seven is: 7.000000

The value of eight and a half is: 8.500000

Can you find all possible compilation errors and logic errors? Can you fix them to print the same result as the expected output? Before you use your compiler, try to find the errors only by manual code analysis.

CODE (with corrections):
#include <iostream>
put #include <iomanip> to be able to use for std::setprecision to be able to display 7.000000 and 8.500000 or else it will just display 7 and 8.5 only

using namespace std;

int main()

{
// add cout << fixed << std::setprecision(6); to display 7.000000 and 8.500000

cout<<"The value of seven is: "<< 7 0; // add a point point between 7 and 0 then type 7.000000 << std::endl for it to display "7.000000" and to make space for the next line of code.

cout<<"The value of eight and a half is: "<<8.5; // type 8.500000 to display "8.500000"

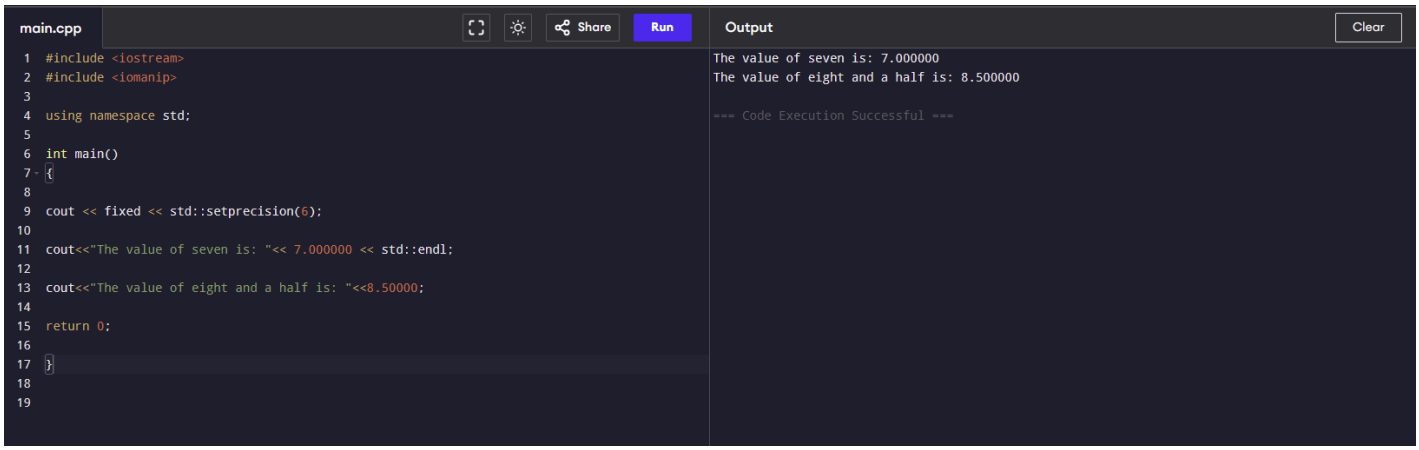
return 0;

}

CODE (corrections applied):

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

using namespace std;

int main()

{

cout << fixed << std::setprecision(6);

cout<<"The value of seven is: "<< 7.000000 << std::endl;

cout<<"The value of eight and a half is: "<<8.50000;

return 0;

}
```

RESULT:



Example 3: The following program has an output of:

The value of half is: 0.500000

The value of Pi is: 3.141593

Can you find all possible compilation errors and logic errors? Can you fix them to print the same result as the expected output? Before you use your compiler, try to find the errors only by manual code analysis.

add preprocessor directive which is #include<iostream> and #include <iomanip>(so we can use std::cout << std::fixed <<std::setprecision; for the floats that need to be displayed)

int main()

{

float halfValue = 0.6; change the value to 0.5

float piValue = 3.141 592 65; //no spaces between the numbers

// add std::cout << std::fixed <<std::setprecision; to display 0.500000 and The value of Pi is: 3.141593

cout<<"The value of half is: "<< half Value; //no spaces between half and Value, add "std::" at the beginning and add "<<std::endl;" at the end so the text isn't at the same line

cout<<"The value of Pi is: "<<pi_Value; //remove the underscore and "std::" at the beginning

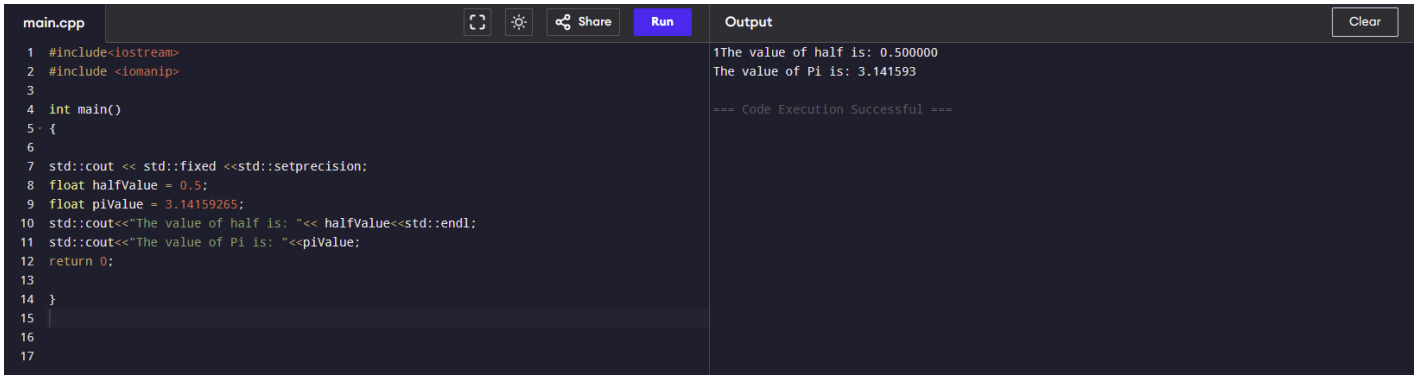
return 0;

}

CODE (corrections applied):

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

int main()

{

std::cout << std::fixed <<std::setprecision;

float halfValue = 0.5;

float piValue = 3.14159265;

std::cout<<"The value of half is: "<< halfValue<<std::endl;

std::cout<<"The value of Pi is: "<<piValue;

return 0;

}
```

RESULT:



***Example 4**: Sample program for Adding Two Integers*

CODE(with corrections)

#include <iostream>

```
int main()

{

int integer1, integer2, sum; /*declaration */

cout<<“Enter first integer: \n” ;  /* prompt */  // string was coded incorrectly and there is no “std::”

cin>>integer1 ;              /* read an integer */ // there is no “std::”

cout<<“Enter second integer: \n” ; /* prompt */ // string was coded incorrectly and there is no “std::”

cin<<integer2;               /* read an integer */ // its suppose to be cin>>integer2; because its an input

sum = integer1 + integer2;          /* assignment of sum */

cout<<“Sum is : "<<sum;       /* print sum */ // string was coded incorrectly and there is no “std::”


return 0;   /* indicate that program ended successfully */

}
```

Sample Output:

Enter first integer

45

Enter second integer

72

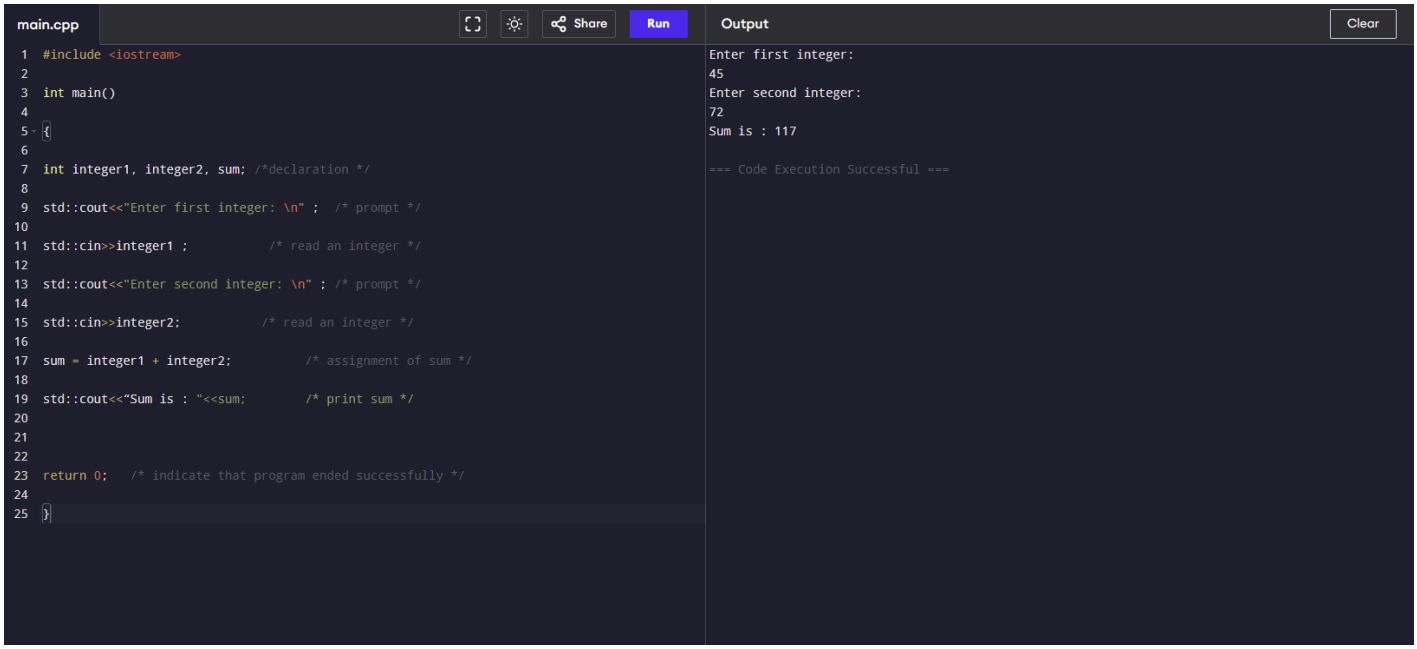
CODE(corrections applied)

```
#include <iostream>

int main()

{

int integer1, integer2, sum; /*declaration */

std::cout<<"Enter first integer: \n";  /* prompt */

std::cin>>integer1 ;          /* read an integer */

std::cout<<"Enter second integer: \n"; /* prompt */

std::cin>>integer2;          /* read an integer */

sum = integer1 + integer2;          /* assignment of sum */

std::cout<<"Sum is : "<<sum;       /* print sum */
```

return 0; / indicate that program ended successfully */*

}

RESULTS:



Supplementary Activity

1. Take a look at the code below: it assigns two integer values, manipulates them and finally outputs the result and bigresult variables. The problem is that the manipulations have been described using natural language, so the code is completely useless now. Act as an intelligent (naturally!) compiler and translate the formula into a real "C" code notation. Test your code using the data provided.

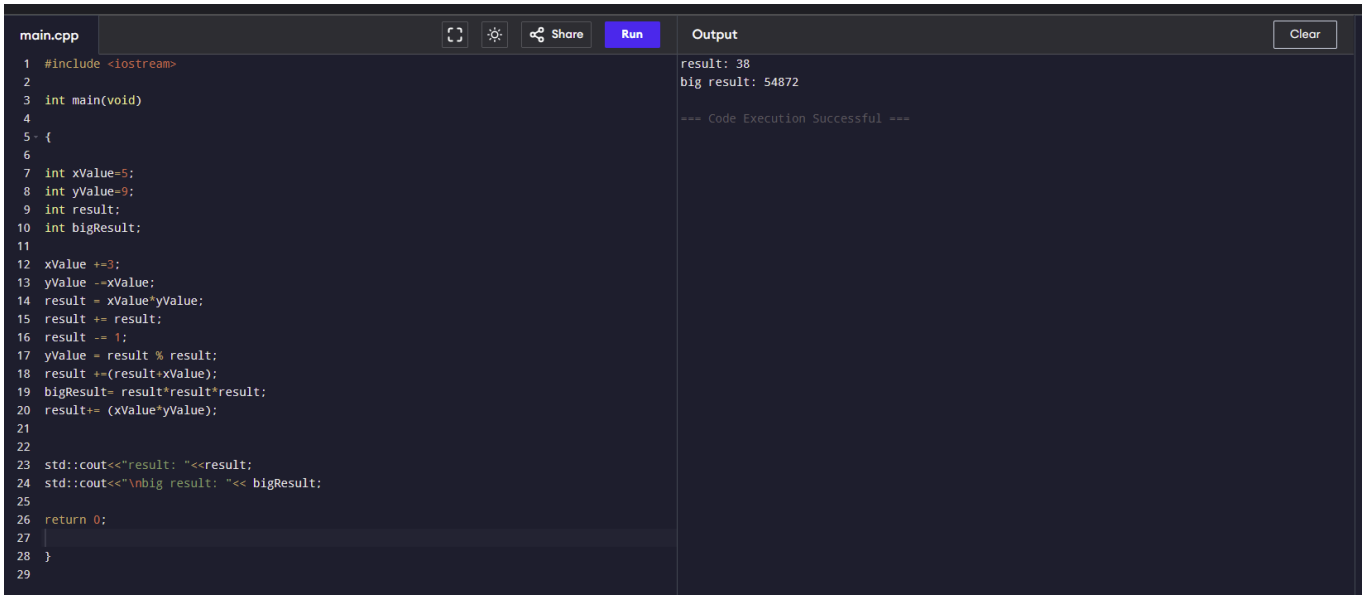
```
#include <iostream>
using namespace std;
int main(void)
{
int xValue=5;
int yValue=9;
int result;
int bigResult;
 /*
increment xValue by 3
decrement yValue by xValue
multiply xValue times yValue giving result
increment result by result
decrement result by 1
assign result modulo result to yValue
increment result by result added to xValue
assign result times result times result to bigResult  increment result by xValue times yValue
*/

cout<<"result: "<<result;
```

cout<<"big result: "<< bigResult;
return 0;
}

CODE:

```
#include <iostream>

int main(void)

{

int xValue=5;
int yValue=9;
int result;
int bigResult;

xValue +=3;
yValue -=xValue;
result = xValue*yValue;
result += result;
result -= 1;
yValue = result % result;
result +=(result+xValue);
bigResult= result*result*result;
result+= (xValue*yValue);


std::cout<<"result: "<<result;
std::cout<<"\nbig result: "<< bigResult;

return 0;

}
```

RESULTS:



2. Complete the program below. Compute the accrued amount of money with a starting value of 100 and an annual interest rate of 1.5%. Compute and print the results for first three years. Your version of the program must print the same result as the expected output for every year. Compute each annual value on the basis of the previous year's value.

```
#include <iostream>
using namespace std;
int main()
{
float startValue = 100;
float interestRate = 0.015;
float firstYearValue;
float secondYearValue;
float thirdYearValue;

/* Your code */

cout<<"After first year: "<<firstYearValue;
cout<<"After second year: "<<secondYearValue; cout<<"After third year: "<<thirdYearValue;
return 0;
}
```

Example output

After first year: 101.500000
After second year: 103.022499
After third year: 104.544998

CODE:

```
#include <iostream>

int main()

{

float startValue = 100;
float interestRate = 0.015;
float firstYearValue;
float secondYearValue;
float thirdYearValue;

firstYearValue = startValue + (startValue * interestRate);
secondYearValue = firstYearValue + (startValue * interestRate);
thirdYearValue = secondYearValue + (startValue * interestRate);

std::cout<<"After first year: "<<firstYearValue;
std::cout<<"\nAfter second year: "<<secondYearValue;
std::cout<<"\nAfter third year: "<<thirdYearValue;

return 0;

}
```

RESULTS:

main.cpp

```cpp
#include <iostream>


int main()

{

float startValue = 100;
float interestRate = 0.015;
float firstYearValue;
float secondYearValue;
float thirdYearValue;

firstYearValue = startValue + (startValue * interestRate);
secondYearValue = firstYearValue + (startValue * interestRate);
thirdYearValue = secondYearValue + (startValue * interestRate);

std::cout<<"After first year: "<<firstYearValue;
std::cout<<"\nAfter second year: "<<secondYearValue;
std::cout<<"\nAfter third year: "<<thirdYearValue;

return 0;

}

```

Output

```
After first year: 101.5
After second year: 103
After third year: 104.5

=== Code Execution Successful ===
```

Conclusion

Fixing the code was very difficult, but after a lot of trial and error, I managed to fix all the code and saw all the errors that caused the code to error in the first place. It was very stressful, but it gave me more awareness and understanding of why these kinds of things happen.

From these two activities, I learned how to translate real-world logic into proper C++ code, especially by understanding how variables are manipulated step-by-step. I experienced a lot of trial and error while fixing and creating the code for these exercises, but I managed to persevere and gain more knowledge that will enhance my coding skills for future tasks.

Assessment Rubric

Rubric for SO 7 (6)

Criteria	Ratings						Pts
SO 7 PI 1 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory | Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory | Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor | Relies on classroom instruction only	1 pts Very Poor | No initiative or interest in acquiring new knowledge	6 pts
SO 7 PI 2 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Completes an assigned task independently and practices continuous improvement	5 pts Good | Completes an assigned task without supervision or guidance	4 pts Satisfactory | Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory | Requires detailed or step-by-step instructions to complete a task	2 pts Poor | Shows little interest to complete a task independently	1 pts Very Poor | No interest to complete a task independently	6 pts
SO 7 PI 3 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good | Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory | Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory | Apply the gathered information to formulate the problem	2 pts Poor | Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor | Gather information from a variety of sources	6 pts
SO 7 PI 4 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good | Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory | Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory | Shows some creative ways to solve the problem	2 pts Poor | Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor | Ideas are copied or restated from the sources consulted	6 pts

Total Points: 24