Problem 1 a *(10 points)* Write a complete `C++` program that asks the user to enter a positive number `n` and prints a horizontal line with `n` copies of `n`. There should be no spaces between the output characters. There is no need for your program to check that the user enters valid input. Here is an example of how the program should run:

```
Enter a positive number: 6
666666
```

Answer:

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Enter a positive number:  ";
   cin >> n;
   for (int c = 1; c <= n; c++) cout << n;
   cout << endl;
   return 0;
}
```

Award 1 point partial credit for each of the following. Each item gets 1 point if it is completely correct and otherwise gets 0:

- include and using
- the main() line
- declare n
- prompt
- cin instruction
- for keyword
- for loop control
- looped cout statement
- newline outside the loop
- return 0

If a program follows a different (but reasonable) plan. Try to award partial credit for meeting similar goals in the program.

Problem 2 a *(10 points)* Write C++ statements to carry out the following tasks. Do not write complete programs. Each answer should be at most 4 lines of C++. Assume the following variables have been declared and initialized with positive values.

```
int x, y;
```

(a) Ask the user to enter a positive value for x, input the user's choice. Do not check it is positive.

Answer:

```
cout << "Enter a positive value for x: ";
cin >> x;
```

(b) If x is not positive print the message Error.

Answer:

```
if (x <= 0) cout << "Error\n";
```

(c) Print the remainder when x^4 is divided by 5.

Answer:

```
cout << (x * x * x * x) % 5 << endl;
```

(d) Repeatedly multiply y by x until it is larger than 999.

Answer:

```
while (y < 1000) y = y * x;
```

(e) Print the last digit of y.

Answer:

```
cout << y % 10 << endl;
```

Each part gets 2 points if it is correct and 0 if it is wrong.

Problem 3 a *(10 points)* Consider the following C++ program.

```
int main() {
    int x = 11, y = 9, z = 11;
    string name = "Freddy";

    cout << ((x/3) * 10.0) / 4 << endl;                    // line (a)
    cout << (17 % 5) % 2 + 20 << endl;                     // line (b)
    if (x > y && x < z)                                    // line (c)
      cout << x << endl;
    else cout << "Hello" << endl;
    if (y > y || name == "Freddy")                         // line (d)
      cout << "Goodbye" << endl;
    else cout << 12 << endl;
    for (x = 1; x <= 3; x++)                               // line (e)
      for (y = 4; y < 6; y++)
         cout << x << y;
    cout << endl;
    return 0;
}
```

(a) What is the output from the instruction beginning on line (a)?

Answer:

```
7.5
```

(b) What is the output from the instruction beginning on line (b)?

Answer:

```
20
```

(c) What is the output from the instruction beginning on line (c)?

Answer:

```
Hello
```

(d) What is the output from the instruction beginning on line (d)?

Answer:

```
Goodbye
```

(e) What is the output from the instruction beginning on line (e)?

Answer:

```
141524253435
```

Each part gets 2 points if it is correct and 0 if it is wrong. Parts a and e can get 1 point partial credit if partially correct.

Problem 4 a *(10 points)* In this problem, you will write a complete `C++` program that asks the user to enter a positive odd number `n` and prints a solid triangular arrow made with `*` symbols that has `n` rows. You should not check that the user's response is legal and no partial credit will be given for doing so. Here is an example of how the program should run:

```
Enter a positive odd number: 7
*
**
***
****
***
**
*
```

(a) Give a formula (in terms of `n`) for the number of columns that should be printed. For example, if `n` is 7, the number of columns is 4 as in the example.

`n / 2 + 1`

(b) Write the complete `C++` program below.

Answer:

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Enter a positive odd number:  ";
   cin >> n;
   for (int r = 1; r <= n; r++) {
      for (int c = 1; c <= n / 2 + 1; c++) {
         if (c <= r && c <= (n + 1) - r) cout << "*";
      }
      cout << endl;
   }
   return 0;
}
```

Part a gets 2 points

For part b, award 1 point partial credit for each of the following. Each item gets 1 point if it is completely correct and otherwise gets 0:

- The answer looks like a complete C++ program (eg has include, esing, return etc.).
- Interaction with user to read n
- Outer loop control
- Inner loop control
- `c <= r` part
- && other condition
- ouput * as an inner loop action
- output endl as an outer loop action

If a program follows a different (but reasonable) plan. Try to award partial credit for meeting similar goals in the program.

Another reasonable plan would be two nested loops, one for the top half of the diagram and the other for the bottom half.

Problem 1 b *(10 points)* Write a complete `C++` program that asks the user to enter a positive number `n` and prints a vertical line with `n` copies of the letter `X`. There is no need for your program to check that the user enters valid input. Here is an example of how the program should run:

```
Enter a positive number: 4
X
X
X
X
```

Answer:

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Enter a positive number:  ";
   cin >> n;
   for (int c = 1; c <= n; c++) cout << "X\n";
   return 0;
}
```

Award 1 point partial credit for each of the following. Each item gets 1 point if it is completely correct and otherwise gets 0:

- include and using
- the main() line
- declare n
- prompt
- cin instruction
- for keyword
- for loop control
- looped cout statement
- newline outside the loop
- return 0

If a program follows a different (but reasonable) plan. Try to award partial credit for meeting similar goals in the program.

Problem 2 b *(10 points)* Write `C++` statements to carry out the following tasks. Do not write complete programs. Each answer should be at most 4 lines of `C++`. Assume the following variables have been declared and initialized with positive values.

```
int r, c;
```

(a) Ask the user to enter a non-zero value for `c`, input the user's choice. Do not check it is non-zero.

Answer:

```
cout << "Enter a non-zero value for c: ";
cin >> c;
```

(b) If `c` is zero print the message `Error`.

Answer:

```
if (c == 0) cout << "Error\n";
```

(c) Print the remainder when r^3 is divided by `c`.

Answer:

```
cout << (r * r * r) % c << endl;
```

(d) If `c` is 0, repeatedly enter a new user input value for `c`, until `c` is not 0.

Answer:

```
while (c == 0) {
   cout << "Enter a positive value for c: ";
   cin >> c;
}
```

(e) Print the last two digits of `r`.

Answer:

```
cout << r % 100 << endl;
```

Each part gets 2 points if it is correct and 0 if it is wrong.

Problem 3 b *(10 points)* Consider the following C++ program.

```
int main() {
    int x = 5, y = 10, z = 9;
    string name = "freddy";

    cout << ((x/3) * 10.0) / 4 << endl;                    // line (a)
    cout << (15 % 7) % 2 + 20 << endl;                     // line (b)
    if (x <= y && x <= z)                                  // line (c)
      cout << x << endl;
    else cout << "Hello" << endl;
    if (y > y || name == "Freddy")                         // line (d)
      cout << "Goodbye" << endl;
    else cout << 12 << endl;
    for (x = 8; x <= 9; x++)                               // line (e)
      for (y = 1; y < 3; y++)
         cout << x << y;
    cout << endl;
    return 0;
}
```

(a) What is the output from the instruction beginning on line (a)?

Answer:

```
2.5
```

(b) What is the output from the instruction beginning on line (b)?

Answer:

```
21
```

(c) What is the output from the instruction beginning on line (c)?

Answer:

```
5
```

(d) What is the output from the instruction beginning on line (d)?

Answer:

```
12
```

(e) What is the output from the instruction beginning on line (e)?

Answer:

```
81829192
```

Each part gets 2 points if it is correct and 0 if it is wrong. Parts a and e can get 1 point partial credit if partially correct.

Problem 4 b *(10 points)* In this problem, you will write a complete `C++` program that asks the user to enter a positive number `n` and prints a solid triangular arrow made with `#` symbols that has `n` columns. You should not check that the user's response is legal and no partial credit will be given for doing so. Here is an example of how the program should run:

```
Enter a positive number: 4
#
##
###
####
###
##
#
```

(a) Give a formula (in terms of `n`) for the number of rows that should be printed. For example, if `n` is 4, the number of rows is 7 as in the example.

`2 * n - 1`

(b) Write the complete `C++` program below.

Answer:

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cout << "Enter a positive number:  ";
   cin >> n;
   for (int r = 1; r <= 2* n - 1; r++) {
      for (int c = 1; c <= n; c++) {
         if (c <= r && c <= 2 * n - r) cout << "#";
      }
      cout << endl;
   }
   return 0;
}
```

Part a gets 2 points

For part b, award 1 point partial credit for each of the following. Each item gets 1 point if it is completely correct and otherwise gets 0:

- The answer looks like a complete C++ program (eg has include, esing, return etc.).
- Interaction with user to read n
- Outer loop control
- Inner loop control
- `c <= r` part
- && other condition
- ouput # as an inner loop action
- output endl as an outer loop action

If a program follows a different (but reasonable) plan. Try to award partial credit for meeting similar goals in the program.

Another reasonable plan would be two nested loops, one for the top half of the diagram and the other for the bottom half.