Instructions:

1. The exam will be graded out of 38.
2. Turn off all electronic media and store them under your desk.
3. If you are asked to author code, it must be implemented using C++. Unless we specifically ask for syntactically correct C++ code, we may ignore one or two small syntactical errors such as a missing semi colon.
4. You may use `cout` and `endl` instead of `std::cout` and `std::endl`.
5. You may ask only one question during the examination: "May I go to the washroom?"
6. Asking any other question will result in a deduction of 5 marks from the exam grade.
7. If you think a question is ambiguous, write down your assumptions and continue.
8. Do not leave during first hour or after there are only 15 minutes left.
9. Do not stand up until all exams have been picked up.
10. If a question only asks for an answer, you do not have to show your work to get full marks; however, if your answer is wrong and no rough work is presented to show your steps, no part marks will be awarded.

1. (5 points) Determine any errors (syntactical or logical) in the following program and correct them.

```
include <iostream>

int main();
int median( int a, int b, int c );

int main() {
    std::cout << median( 5, 2, 9 ) << std::endl;
    return 0;
}

// The median of three numbers
// is the middle of the three
median( int a, int b, int c ) {
  if ( ((a <= b) and (b <= c)) or ((c <= b) and (b <= a)) ) {
    return b;
  else if ( ((a <= c) and (c <= b)) or ((b <= c) and (a <= c)) ) {
    return c;
  }

  return a;
}
```

2. (3 points) Two local variables `m` and `n`, both of type `int`, are declared and initialized inside a program. Write code to assign to `m` the sum of the two original values assigned to `m` and `n`, and assign to `n` the product of the two original values assigned to `m` and `n`.

3. (5 points) A local variable `n` is assigned an integer greater than zero. Someone else has written a function with the declaration

```
// Return `true' if `m' is a prime
// number, and `false' otherwise.
bool is_prime( int m );
```

Write and complete the function definition of `main()` for a program that prints out the n^{th} prime number. For example, 2 is the first prime number, 3 is the second, 5 is the third, and so on.

```
int main() {
  int n{};
  std::cout << "Enter a positive integer: ";
  std::cin  >> n;  // You may assume the user enters a positive integer
```

4. (6 points) In this question, you can use `cout` and `endl` without prefixing them by the `std::` namespace. Implement the function `void squiggle( unsigned int n )` that draws a slash (/), followed by n backslashes (\), followed by another slash in the pattern shown here:

   When $n = 0$, the output is:

```
 /
/
```

   When $n = 1$, the output is:

```
/
\
/
```

   When $n = 2$, the output is:

```
/
\
 \
 /
```

   When $n = 3$, the output is:

```
/
\
 \
  \
  /
```

   When $n = 4$, the output is:

```
/
\
 \
  \
   \
   /
```

5. (2 points) What is the output of this program?

```
#include <iostream>
int main();
void f( int &n, int low, int high );

int main() {
  int m{  53 };
  int n{ -37 };
  f( m, 0, 10 );
  f( n, m,  0 );

  std::cout << m << std::endl;
  std::cout << n << std::endl;

  return 0;
}

void f( int &n, int low, int high ) {
  if ( n < low ) {
    n = low;
  } else if ( n > high ) {
    n = high;
  }
}
```

6. (4 points) Noting that `-1`, `-0b1` and `-0x1` all represent a literal -1, consider the following two declarations:

```
short m{ 0b11010101111 };
short n{ -0b1100100110 };
```

First, in the below code, fill in the blanks with the hexadecimal digits that would give `m` and `n` the same values as above:

```
short m{  0x           };
short n{ -0x           };
```

Second, in the below code, fill in the blank with the binary digits that would give `n` the same value, but without the explicit minus sign:

```
short n{ 0b                          };
```

Finally, consider the declaration `short result{n - m};`

Using binary addition as shown in class, show the sequence of sixteen bits that would be stored in the local variable `result` (you don't have to convert the result to decimal).

7. (3 points) Write out the value of `m` in binary after the following declaration and statement. If you get the correct answer, you get full marks; however, if your solution is incorrect, you cannot get part marks if you do not show your intermediate steps.

```
unsigned int m{ (1 << 8) - 1 };
m ^= (m << 4);
```

8. (4 points) You will implement a program that takes an array of `double` that has `N` entries as shown below. Complete the program so that once it is executed, it will print the number of negative numbers, the number of zeros, and the number of positive numbers. For example, one example of possible output is

```
There are 7 negative number(s),
4 zero(s), and 1 positive number(s).
```

In this question, you may use `cout` and `endl` without the `std::` namespace.

```
int main() {
  unsigned int const N{ /* some value... */ };
  double arr[N]{};
  std::cout << "Enter " << N << " floating-point numbers: ";

  for ( unsigned int k{ 0 }; k < N; ++k ) {
    std::cin >> arr[k];
  }









    return 0;
}
```

9. (6 points) Write a function with the declaration below that returns an integer that reverses the decimal digits of the argument `n`, leaving the result negative if the argument is negative. For example, -9 through 9 are left unchanged, 12345 is changed to 54321, 140 is changed to 041 which equals 41, and -10250 is changed to -5201. In the comments of your code, briefly explain how you deal with negative arguments.

```
int reverse( int n );
```