1. Write the number of the definition on the right next to the term it defines.

(a) **reference**_____

(b) **variable**_____

(c) **precondition**_____

(d) **declaration**_____

(e) **parameter**_____

(f) **byte**_____

(g) **parser**_____

(h) **scope**_____

(i) **postcondition**_____

(j) **type**_____

(k) **object**_____

(1) A declaration of an explicit input to a function through which a function can access the arguments passed by name.

(2) A condition that must hold upon exit from a piece of code, such as a function or a loop.

(3) The region of program text (source code) in which a name can be referred to.

(4) A requirement of a function upon its argument that must be true for the function to perform its action correctly.

(5) The specification of a name with its type in a program.

(6) A named object of a given type; contains a value unless uninitialized.

(7) Some memory that holds a value of a given type.

(8) A program that reads a stream of tokens according to a grammar.

(9) The basic unit of addressing in most computers.

(10) A value describing the location of a typed value in memory.

(11) A set of possible values and a set of operations (for an object).

2. Construct `C++` loop that will

(a) lead to an overflow error.

(b) cause a range error.

3. Write a `C++` assignment statement that will cause a narrowing error from

(a) an `int` to a `char`.

(b) a `double` to an `int`.

4. Consider the absolute value function defined below:

```
double abs(double x) { return (x>0) ? x : -x; }
```

Rewrite the function using an `if/else` structure instead of the ternary operator.

5. The function `getPoly()` below is designed to get the coefficients, $a_0, a_1, \ldots, a_n$ of a polynomial $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ of degree n.

```
vector<double> getPoly() {
    double deg{0};
    cout << "\nWhat is the degree of your polynomial? ";
    do {
        cin >> deg;
    } while(deg <= 0 || int(deg) != deg);
    vector<double> coeff(int(deg)+1); //create a vector deg+1 doubles = 0.0
    cout << "\nEnter the coefficients in ascending order: \n";
    for(int i = 0; i < deg+1; ++i) {
        cout << "The coefficient of x^" << i << " = ";
        cin >> coeff[i];
    }
    return coeff;
}
```

(a) What is the purpose of the `do/while` loop?

(b) The function could be improved by changing the type of the variable `deg`. How would you do that?

(c) Suppose the user wishes to enter the polynomial $x^3 - x^2$. Write all the text that would be on the console when this is done.

(d) If each double requires 8 bytes, how many bytes would the polynomial $x^3 + 2x^2 + 3x + 4$ use to store the coefficient vector?

6. The function `poly()` defined below is designed to evaluate a polynomial specified by a vector of coefficients, `coeff`.

```
double poly(vector<double> coeff, double x) {
    double value = coeff[coeff.size()-1];
    for(int i = coeff.size()-1; i > 0; --i) {
        value *= x;
        value += coeff[i-1];
    }
    return value;
}
```

Suppose `coeff` contains {`0,0,-1,1`} and `x = 2`.

(a) What is `coeff.size()`?

(b) Complete the following table of values as the function executes until its `return` is called. Does the function return the proper value `poly`(2) ?

i	value
	1

(c) Repeat part (b) to evaluate `poly`(3) for `poly`$(x) = x^3 - 2x^2 + 3$ where `coeff = {3,0,-2,1}`.

i	value
	1

7. Write a definition for `double secant(vector<double> coeff, double a, double b)` that will call the `poly()` function of problem #6 above and return the slope of the secant line from $(a, p(a))$ to $(b, p(b))$. You can do this in one line.

8. The decimal form of $\frac{1}{13} = 0.076923076923\cdots = 0.\overline{076923}$ has a repetend (the part of the decimal that repeats) of 076923. The program below is designed to find the length of the repetend of a fraction whose numerator is 1 and whose denominator is specified by the user. Most of the code is shown below:

```
// Note, a "short" is a 2 byte integer.
void getDecimal(vector<short>& v, unsigned n);
unsigned findRepetend(vector<short> decimals);

int main() {
    unsigned denominator;
    vector<short> decimalDigits;
    cout << "\nEnter the denominator of your rational number: ";
    while(cin >> denominator) {
        decimalDigits.clear();
        getDecimal(decimalDigits, denominator);
        cout << denominator << " has a repetend of length "
             << findRepetend(decimalDigits);
    }
}
void getDecimal(vector<short>& v, unsigned n) {
    unsigned power10 = 10;
    while(power10/n==0) power10 *= 10;
    v.push_back(power10/n);
    unsigned remainder = power10%n;
    for(int i = 0; i< 101; ++i) {
        remainder *= 10;
        v.push_back(remainder/n);
        remainder = remainder%n;
    }
}
unsigned findRepetend(vector<short> decimalDigits) {
    // supply code for algorithm
}
```

(a) Suppose `d` is a `vector<short>`. Describe what `getDecimal(d, 13)` does, step by step.

(b) On an attached page, write a program to define `findRepetend()`. Use the flow chart at right, it it helps. If the user enters 13, the output should read "`13 has a repetend of length 6`". Remember to use proper syntax and style and to define your variables.