1. Match each of the following terms on the left with its definition on the right.

(a) **variable**	(1) A set of possible values and a set of operations (for an object).
(b) **divide and conquer**	(2) A named unit of code that can be invoked (called) from different parts of a program; a logical unit of computation.
(c) **declaration**	(3) A statement that introduces a new name into a program and sets aside memory for a variable.
(d) **byte**	(4) The problem-solving strategy of taking a large problem and divide it into several little ones.
(e) **constructor**	(5) A named object.
(f) **definition**	(6) A statement that gives a name to an object.
(g) **function**	(7) Some memory that holds a value of a given type.
(h) **type**	(8) A special function that initializes and acquires resources needed for a user-defined object.
(i) **object**	(9) The basic unit of addressing in most computers.

2. Each of the following programs has error(s). Locate the error(s), classify each error as either a syntax error, a logical error, an overflow error, an underflow error, or a narrowing error.:

(a)
```
int average(int a, int b)
{
    return a + b / 2;
}
```

(b)
```
int main() {
   vector<int> iVector(10);
   myvector[10]=5;
}
```

(c)
```
int  main(void) {
    int x{7777.7};
    char c2{7777};
    cout << c2 << " == " << char(x%256);
}
```

(d)
```
int  foo(int x) {
    return(x+1);
}

int  main() {
    cout << foo(INT_MAX);
}
```

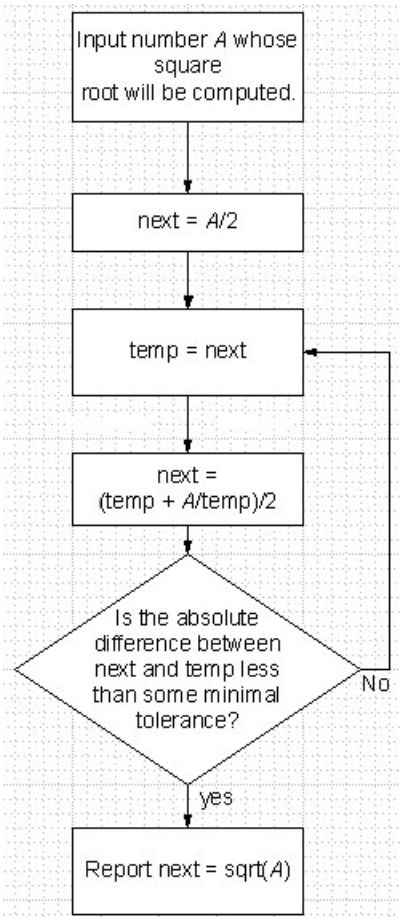
(e)
```
int  main(void) {
    unsigned int a{2};
    unsigned int b{3};
    a -=b;
}
```

3. Explain what the following program does:

```
void foo(int a, int& b) {
  a*=2;
  cout << "\na = " << a;
  b*=2;
}

int main() {
  int x=1, y=3;
  foo(x, y);
  cout << "x=" << x << ", y=" << y;
}
```

4. Write a program that implements the Babylonian algorithm. Use the flow chart at right as a guide, if you like. Remember to use proper syntax and style and to define your variables.



5. Write a C++ program to implement the following pseudo-code for Euclid method of finding the greatest common divisor of `m` and `n`.

```
if m < n, swap(m,n)
while n does not equal 0
   r = m modulo n
   m = n
   n = r
endwhile
output m
```

6. Consider the complete program below:

```
#include "std_lib_facilities.h"

const int MAX{8};

void cycle_sort(vector<int>&);

int main() {
    vector<int> a(MAX);
    int i;

    cout << "enter the elements into array :";
    for (i = 0;i < MAX; i++)     {
        cin >> a[i];
    }
    cycle_sort(a);
    cout << "sorted elements are :\n";
    for (i = 0;i < MAX; i++)  {
        cout << a[i] << " ";
    }
}

/* sorts elements using cycle sort algorithm */
void cycle_sort(vector<int>& a) {
    int item, pos, i, j, k;

    for (i = 0; i < MAX; i++)    {
        item = a[i];
        pos = i;
        do {
            k = 0;
            for (j = 0; j < MAX; j++) {
                if (pos != j && a[j] < item) {
                    k++;
                }
            }
            if (pos != k)  {
                while (pos != k && item == a[k]) {
                    k++;
                }
                swap(a[k],item);
                pos = k;
            }
        } while (pos != i);
    }
}
```

(a) Which line contains a function declaration/prototype?

(b) Which lines contain the definition of this function?

(c) Carefully describe the parameter list of this function.

(d) What is the condition for continuing the do-loop of the function?

(e) What is the condition for ending the for-loop in the body of this do-loop?

(f) Suppose the user enters `1 3 2 4 5 7 8 6`. Trace the values of the various components of the program as the instructions are followed, step by step:

a	i	item	pos	k	j	`pos!=j && a[j]<item`	`pos!=k`	pos!=i
{1 3 2 4 5 7 8 6}	0	1	0	0	0	`false` && `false`	`false`	`false`