CS 7B Midterm 1 - Spring 2018 - Midterm 1 3/21/18

1. Write the number of the definition on the right next to the term it defines.
   (a) **class**______
   (b) **constructor**________
   (c) **container**________
   (d) **copy**_____
   (e) **invariant**________
   (f) **overload**________
   (g) **reference**________
   (h) **pointer**________
   (i) **scope**________
   (j) **byte**________
   (k) **type**________
   (l) **move**________

   (1) An operation that makes two objects have values that compare equal..
   (2) The region of program text (source code) in which a name can be referred to.
   (3) A user-defined type that may contain data members, function members, and member types.
   (4) An operation that initializes an object. Typically establishes an invariant and often acquires resources needed for an object to be used (which are then typically released by a destructor).
   (5) (1) a value used to identify a typed object in memory; (2) a variable holding such a value.
   (6) (1) a value describing the location of a typed value in memory; (2) a variable holding such a value.
   (7) Define two functions or operators with the same name but different argument (operand) types.
   (8) an operation that transfers a value from one object to another, leaving behind a value representing "empty."
   (9) Something that defines a set of possible values and a set of operations for an object.
   (10) Something that must be always true at a given point (or points) of a program; typically used to describe the state (set of values) of an object or the state of a loop before entry into the repeated statement.
   (11) The basic unit of addressing in most computers.
   (12) An object that holds elements (other objects).

2. Consider the following code fragment for the user-defined type, `Token`:

```
struct Token{
    char kind;
    double value;
    string varname;
    Token(char ch) :kind(ch), value(0) { }
    Token(char ch, double val) :kind(ch), value(val) { }
    Token(char ch, string val) :kind(ch), varname(val) { }
};
```

   (a) What `kind` values are used to declare a variable?

   (b) Describe in detail the ways in which the constructors for `Token` are used.

   (c) Describe how to change `Token` from a `struct` to a `class`.

3. Consider the following code fragment for the user-defined type, `Token_stream`

```
struct Token_stream {
    bool full;
    Token buffer;
    istream & str;
    Token_stream(istream & arg) : str(arg), full(0), buffer('␣') { }
    Token get();
    void unget(Token t) { buffer=t; full=true; }
    void ignore(char);
};
```

(a) Describe the variable `buffer`. What is it? What is it for? Give an example of how it is used.

(b) Describe the variable `str`. What is it? What is it for? Give an example of how it is used.

4. Consider the following code fragment for the member function `get()`.

```
Token Token_stream::get() {
    if (full) { full=false; return buffer; }
    char ch;
    str >> ch;
    if (! str) return(Token(quit));
    switch (ch) {
    case '(':   case ')':        case '+':        case '-':        case '*':
    case '/':   case '%':        case ';':        case '=':        case ',':
        return Token(ch);
    case '.':   case '0':        case '1':        case '2':        case '3':
    case '4':   case '5':        case '6':        case '7':        case '8':
    case '9':
    {   str.unget();
        double val;
        str >> val;
        if (! str) error("Bad␣token");
        return Token(number,val);
    }
    default:
        if (isalpha(ch) || ch == '_') {
            string s; s += ch;
            while(str.get(ch) &&
                 (isalpha(ch) || isdigit(ch) || ch == '_'))
                s += ch;
            str.unget();
            if (! str) error("Bad␣token");
            if (s == "let") return Token(let);
            if (s == "const") return Token(constant);
            if (s == "reset") return Token(reset);
            if (s == "sqrt") return Token(sqroot);
            if (s == "pow") return Token(power);
            if (s == "help") return Token(help);
            if (s == "quit" || s == "exit")
                return Token(quit);
            return Token(name,s);
        }
```

```
            error("Bad token");
            return Token(' ');
        }
}
```

(a) What does `get()` get if `full==true`?

(b) What is `str` here?

(c) What is the purpose of `str.unget()` on line 14?

(d) Describe what happens in the `default case`. How does it provide for the declaration of a new variable? How does handle a built-in function like `pow()`? How does it recognize an existing variable in `symbol_table`?

5. Rewrite the calculator program to incorporate the variable

   `istream& str;`

   as a member variable of `Token_stream`, rather than passing it from function to function by reference. Email your complete working code to ghagopian@collegeofthedesert.edu.

6. Consider the following code fragment for handling `Variable`s in the calculator.

```
struct Variable {
    string name;
    double value;
    bool immutable;
    Variable(string n, double v, bool b) :
        name(n), value(v), immutable(b) { }
};

//  The active variables.
class Symbol_table {
    vector<Variable> names;
public:

double get(string s) {
    for (int i = 0; i<int(names.size()); ++i)
        if (names[i].name == s) return names[i].value;
    error("get: undefined name ",s);
        return 0.0;
}

void set(string s, double d)
{   for (int i = 0; i<=int(names.size()); ++i)
        if (names[i].name == s) {
            names[i].value = d;
            return;
        }
    error("set: undefined name ",s);
}
```

(a) Describe the constructor for a `Variable`. How does it work?

(b) What is `Symbol_table`. What purpose does it serve?

(c) Why does this `get()` function not collide with `Token_stream`'s `get()` function?

(d) Describe in detail how `get()` works and what its purpose is.

(e) Describe in detail how `set()` works and what its purpose is.

(f) Could these `get()` and `set()` functions be made member functions of the `struct Variable`? Discuss.