This assignment briefly covers two topics, graph operations and compilers, using the design and implementation of a simple graph calculator. Please refer to the previous homework assignment and the syllabus for important instructions on submission and policies and ask for clarifications if needed. Submit your solutions to the *non-programming questions* in **neat** hardcopy form, being sure to include your name. Sloppy work may be returned without grading, earning a 0. Package and submit your solutions to the *programming questions*, and any other electronic components of your work, as in the previous assignment. Reminder: Submit only source code, documentation, and related files (Makefile, README) as a single `.tgz` packaged file using the proper file-naming convention. All electronic submissions must be made using the interface at `http://cs.umaine.edu/~chaw/u/` only. *See Question 9 for an important note.*

1. (10 pts.) *Well-known graphs.* The notation K_n denotes the *complete* graph on n vertices while $K_{m,n}$ denotes the *complete bipartite* graph with m vertices on one side and n on the other. P_n denotes a *path* graph on n vertices, C_n denotes an n-vertex *cycle*, W_n denotes an n-vertex *wheel*, and S_n denotes an n-vertex *star*.

   Depict K_n, $K_{n-1,n+1}$, P_n, C_n, W_n, and S_n for $n = 3, 5, 10$ in a manner that best illustrates their structure.

2. (20 pts.) *Graph operators.* Given graphs $G = (U, E)$ and $H = (V, F)$, we may define additional graphs based on the following operators. All graphs in this assignment are *undirected graphs without loops and without multiple edges between the same pair of vertices.* Therefore edges (a, b) and (b, a) are identical, and $a \neq b$ for any edge (a, b).

   Complement $\bar{G} = (U, \{(u, u') \mid u, u' \in U, (u, u') \notin E\})$.

   Direct sum $G \oplus H = (U \cup V, E \cup F)$.

   Join $G—H = (U \cup V, E \cup F \cup \{(u, v) \mid u \in U, v \in V\})$.

   Direct product $G \otimes H = (U \times V, \{((u, v), (u', v')) \mid (u, u') \in E, (v, v') \in F\})$.

   Cartesian product $G \,\square\, H = (U \times V, \{((u, v), (u', v)) \mid (u, u') \in E, v \in F\} \cup \{((u, v), (u, v')) \mid u \in E, (v, v') \in F\})$.

   Strong product $G \boxtimes H = (U \times V, E_\otimes \cup E_\square)$ where $G \otimes H = (U \times V, E_\otimes)$ and $G \,\square\, H = (U \times V, E_\square)$.

   Odd product $G \triangle H = (U \times V, \{((u, v), (u', v')) \mid (u, u') \in E$ exclusive or $(v, v') \in F\})$.

   Lexicographic product $G \circ H = (U \times V, \{((u, v), (u', v')) \mid (u, u') \in E$ or $u = u'$ and $(v, v') \in F\})$.

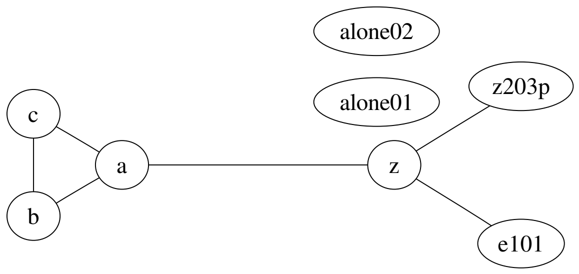
   Consider the set Z of the graphs of Question 1 for $n = 5$. For each pair of graphs $(G, H) \in Z \times Z$, depict $G \cdot H$ replacing $\cdot$ with each of the binary graph operators above, along with $\bar{G}$ for all $G \in Z$.

3. (10 pts.) Depict $P_2 \circ P_2 \circ \cdots \circ P_2$ where the product is applied n times, for $n = 0, 1, 2, 3, 4$, *as neatly as possible.*

4. (10 pts.) Repeat Question 3 for P_2—P_2—$\cdots$—P_2.

5. (10 pts.) *Linear representation.* We define a linear representation of graphs, illustrated by the following example.

```
c--a z--a z
alone01 z--e101 z203p--z
alone02 b--c a--b
```



The input consists of whitespace-separated tokens. Tokens represent either edges or vertices. A vertex token consists simply of an identifier that follows the conventions of C. An edge token has the format u`--`v where u and v are vertex identifiers. A vertex may be introduced either directly, using a vertex token, or indirectly, by its use as an endpoint of an edge (or both).

Provide linear representations for each of the graphs of Question 2, labeling them clearly. Also provide a general form for linear representations of the graphs of Questions 3 and 4.

6. (10 pts.) *Normalized representation.* A linear representation of a graph is in normal form if

   - there is no whitespace other single spaces separating adjacent tokens;
   - no vertex identifier appears in both an edge token and a vertex token;
   - the vertices of each edge are listed lexicographically; and
   - all the tokens are sorted lexicographically.

   For example, the following is a normalized version of the linear representation of Question 5.

```
a--b a--c alone01 alone02 a--z b--c e101--z z--z203p
```

   Convert each of the linear representations of Question 5 into normal form.

7. (10 pts.) *Graph calculator.* We define a small language for creating and manipulating graphs. The language defines the following families of *graph constants*, based on the definitions in Question 1.

symbol	code	example	graph
K_n	`.Kn`	`.K5`	complete graph with 5 vertices
$K_{m,n}$	`.Km,n`	`.K3,2`	complete (3,2)-bipartite graph
P_n	`.Pn`	`.P4`	path with 4 vertices
C_n	`.Cn`	`.C7`	cycle with 7 vertices
W_n	`.Wn`	`.W6`	wheel with 6 vertices
S_n	`.Sn`	`.S13`	star with 13 vertices

The language also includes *graph variables* using the lexical conventions of C. Variables are implicitly declared by their first use, and defined using assignment statements of the form '`varname = expression;`'. A variable that is used in an expression prior to its definition is automatically initialized to the empty graph. The expression on the right-hand side of an assignment statement is evaluated fully before any change to the variable on the left-hand side is made. There is also a *print statement* with syntax '`print varname;`' that prints (to standard output) a normalized representation of the graph that is the value of the given variable. The variable name in the print statement may be replaced by an expression (defined below), in which case the graph resulting from the evaluation of the given expression is printed in normalized form.

Expressions in the language are built using the *operators* of Question 2, along with parentheses for grouping. The default operator precedence is determined by three classes: The first class, with highest precedence (tightest binding) contains only the unary complement operator. The next precedence class is composed of the five multiplicative operators that have the word 'product' in their names. The last precedence class is composed of direct sum and join.

prec. class	symbol	code	example	operator
1	$\bar{}$	`-`	`- G`	complement
2	$\otimes$	`(X)`	`G (X) H`	direct product
	$\square$	`[ ]`	`G [ ] H`	Cartesian product
	$\boxtimes$	`[X]`	`G [X] H`	strong product
	$\triangle$	`(^)`	`G (^) H`	odd product
	$\circ$	`(o)`	`G (o) H`	lexicographic product
3	$\oplus$	`(+)`	`G (+) H`	direct sum
	—	`---`	`G --- H`	join

Expressions may also include *graph literals* composed of the linear notation of Question 5 enclosed in square brackets. For example, the example graph of Question 5 may be assigned to a variable `Q5` as follows:

```
Q5 = [c--a z--a z alone01 z--e101 z203p--z alone02
     b--c a--b];
```

Whitespace may be omitted, and extra whitespace introduced, without any change in semantics when the result is unambiguous. For example, the first line of the input appearing below may be replaced with '`G101=.K3(+).K4(+).W7;`'.

List the output of the calculator on the following input:

```
G101 = .K3 (+) .K4 (+) .W7;
print G101;
G102 = -G101 [ ] .K3,2; G201 = .K5 (^) .K3;
G103 = G101 (o) G102 --- .K5;
G104 = G101 (o) (G102 --- .K5);
print G102; print G103; print G104; print G201;
G101 = G101 [ ] G101; print G101;
G5 = [ a--b c--d e--f b--f d--f g ];
G5 = G5 (+) G5; print G5;
```

8. (20 pts.) Provide input for the graph calculator of Question 7 to generate each of the graphs of Question 2, naming each appropriately. *Try to provide as brief an input as possible.* Similarly, provide a method for generating graph-calculator input that yields the graphs of Question 3 and 4, for a value of n provided as input to your method.

9. (100 pts.) Implement the graph calculator of Question 7 using *flex*, *bison*, and *C* on *gandalf.* Ensure that the source code you submit compiles cleanly and runs correctly on gandalf. Do not submit anything other than source code and associated documentation, packaged as outlined in class.

   Special due date. Answers to this question of this assignment (only) may be submitted until *2010-03-18 2:05 p.m.* **provided** a good submission for the rest of the questions is made before the regular due date on the first page.