1. (10 points) Give the output for the following program, assume the user types: 449\n.

```
%%
"4"          { return 260; }
"44"         { return 261; }
[0-9]+       { return 262; }
.            { return -1;  }
\n           { return 263; }
%%
int yywrap() { return 1; }
```

```
#include <iostream>
int yylex();
int main() {
  int token = yylex();
  while ( token ) {
    std::cout << "token: " << token << std::endl;
    token = yylex();
  }
}
```

---

2. (5 points)

   (a) What is gramarware?

   (b) What is grammarware engineering?

3. (15 points)

   (a) Show that the following grammar is ambiguous.

   (b) Insert semantic actions to count and print, after each newline, the number of parentheses entered by the user. Use the $$ and $n attributes, with no global variables.

```
%{
#include <iostream>
extern int yylex();
void yyerror(const char * msg) { std::cout << msg << std::endl; }
%}
%token CR LPAR RPAR
%%
lines   : lines expr CR
          { std::cout << "accept" << std::endl; }
        | { ; }
        ;
expr    : expr expr
        | LPAR expr RPAR
        | LPAR RPAR
        ;
```

```
extern int yyparse();
int main() {
  yyparse();
}
```

4. (15 points) For the flex and bison specification listed below:

   (a) Write a sentence that describes the strings accepted by the bison spec below?

   (b) Insert semantic actions so that line #10 in the bison spec prints the sum of the numbers entered. You must use the $$ and $n attributes, and you may not introduce any additional variables into the code. Sample output might include:

```
11112222
sum is: 12

sum is: 0
```

```
1  %{
2  #include "parse.tab.h"
3  %}
4  %%
5
6  "1"         { return ONE; }
7  "2"         { return TWO; }
8  "\n"        { return CR; }
9  .           { ; }
10
11 %%
12 int yywrap() { return 1; }
```

```
1  %{
2  #include <iostream>
3  extern int yylex();
4  void yyerror(const char * msg) { std::cout << msg << std::endl; }
5  %}
6  %token ONE TWO CR
7  %%
8
9  lines    : lines expr CR
10            { std::cout << "accept" << std::endl; }
11         | { ; }
12         ;
13
14 expr     : ONE expr TWO
15         |
16         ;
```

```
1  extern int yyparse();
2  int main() {
3    yyparse();
4  }
```

5. (5 points) Give the output for the following program.

```
#include <iostream>
#include <vector>

class Number {
public:
  Number() : number(0) { }
  Number(int n) : number(n) { }
  Number(const Number& a) : number(a.number) { }
private:
  int number;
};
int main() {
  std::vector<Number> vec;
  vec.reserve(2);
  vec.push_back( 12 );
  vec.push_back( 14 );
  vec.push_back( 16 );
  std::cout << vec.size() << std::endl;
  std::cout << vec.capacity() << std::endl;
}
```

---

6. (10 points) For the following program:

   (a) Give the output, if any.

   (b) Does the program crash? If so, how would you fix it?

```
#include <iostream>
#include <string>

class Student {
public:
  Student(const char* s) : name(new std::string(s)) { }
  ~Student() { delete name; }
  const std::string& getName() const         { return *name; }
  void               setName(const char* n) { (*name) = n;  }
private:
  std::string * name;
};

int main() {
  Student student("Goku"), teacher = student;
  teacher.setName("Kakarot");
  std::cout << teacher.getName() << std::endl;
  std::cout << student.getName() << std::endl;
}
```

7. (10 points) Give the output for the following program.

```
#include <iostream>
class Number {
public:
  Number()                { std::cout << "default" << std::endl;     }
  Number(float)           { std::cout << "convert" << std::endl;     }
  Number(const Number&)   { std::cout << "copy" << std::endl;        }
  ~Number()               { std::cout << "destructor" << std::endl; }
  Number& operator=(const Number&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
};
class Pokemon {
public:
  Pokemon(int cp, int hp) {
    combatPower = cp;
    hitPoints = hp;
  }
private:
  Number combatPower;
  int hitPoints;
};
int main() {
  Pokemon* chansey = new Pokemon(400, 376);
}
```

---

8. (10 points) Give the output for the program below.

```
#include <iostream>
class A {
public:
  ~A() { std::cout << "Destroy A" << std::endl; }
  int getNumber() const { return 44; }
  virtual int incrNumber(int number) { return 99+number; }
};
class B : public A {
public:
  ~B() { std::cout << "Destroy B" << std::endl; }
  int getNumber() const { return 33; }
  virtual int incrNumber(int number) { return 17+number; }
};
int main() {
  A* x = new B;
  std::cout << x->getNumber() << std::endl;
  std::cout << x->incrNumber(1) << std::endl;
  delete x;
}
```

9. (20 points) Class Pokemon violates the rule of three. Write member functions so that Pokemon conforms to the rule of three.

```
#include <iostream>
#include <cstring>

class Pokemon {
public:
  Pokemon(const char* n,  int cp) : name(new char[strlen(n)+1]) {
    strcpy(name, n);
  }
  const char* getName() const { return name; }
private:
  char* name;
};
int main() {
  const Pokemon snore("Snorlax", 2310);
  std::cout << snore.getName() << std::endl;
}
```