Purpose:

To go over:

- Lex/Flex
- YACC/Bison

Overview:

Write a C++ program that:

- uses a Lex/Flex tokenizer to recognize the words in the laundry grammar
- uses a YACC/Bison grammar to parse the laundry grammar. It keeps track of what exactly was washed and dried in a data-structure that it passes from leaves to the root.
- The root takes the data-structure and prints the what laundered: how washed and how dried.

Grammar:

The grammar is the same as before:

1. S -> Wash period Dry period
2. Wash -> machine wash Temp What
3. Wash -> hand wash What
4. What -> Type
5. What -> ItemList
6. Type -> lights|darks|delicates
7. ItemList -> Item ItemList
8. ItemList -> Item
9. Temp -> hot|warm|cold
10. Item -> trousers|shirts|underwear|sheets
11. Dry -> DHow dry
12. DHow -> line|tumble

Support Output:

It outputs a summary of what was washed and dried, and how it was laundered:

```
$ ./laundryLang
Please enter an expression: Machine wash hot underwear sheets. Tumble dry.
Machine wash underwear in hot water.  Tumble dry.
Machine wash sheets in hot water.  Tumble dry.
$ ./laundryLang
Please enter an expression: Machine wash cold shirts trousers. Line dry.
Machine wash trousers in cold water.  Line dry.
Machine wash shirts in cold water.  Line dry.
$ ./laundryLang
Please enter an expression: Hand wash delicates. Line dry.
```

```
Hand wash delicates   Line dry.
$ ./laundryLang
Please enter an expression: Cannot parse this.
syntax error, sorry!
```

The Assignment:

1. **Please copy-and-paste the following files:**
2. **Makefile**

3.
   ```
# ----------------------------------------------------------------------- #
# ---                                                               --- #
# ---              Makefile                                         --- #
# ---                                                               --- #
# ---         This file details the dependencies upon the source, object         --- #
# ---     and executable files of the laundry language program.                  --- #
# ---                                                               --- #
# ---     ----     ----     ----     ----     ----     ----     ----     ----     --- #
# ---                                                               --- #
# ---     Version 1a              2017 October 25              Joseph Phillips          --- #
# ---                                                               --- #
# ----------------------------------------------------------------------- #

laundryLang             : laundryLang.tab.o laundryLang.o
                        g++ -o $@ laundryLang.tab.o laundryLang.o -g

laundryLang.o           : laundryLang.cpp laundryLang.h laundryLang.tab.h
                        g++ -c laundryLang.cpp -g

laundryLang.tab.o       : laundryLang.tab.c laundryLang.h laundryLang.tab.h
                        g++ -c laundryLang.tab.c -g

laundryLang.cpp                 : laundryLang.lex
                        flex -o $@ laundryLang.lex

laundryLang.tab.c       : laundryLang.y
                        bison -d laundryLang.y --debug --verbose

laundryLang.tab.h       : laundryLang.y
                        bison -d laundryLang.y --debug --verbose

clean                   :
```

```
            rm laundryLang.tab.? laundryLang.cpp laundryLang.o \
              laundryLang laundryLang.output
```

4. laundryLang.h

5.
```
/*-------------------------------------------------------------------------*
 *---                                                                  ---*
 *---              laundryLang.h                                       ---*
 *---                                                                  ---*
 *---       This file includes files, defines types and declares      ---*
 *---     variables and functions used in common for both the lex/flex          ---*
 *---     and YACC/Bison source files.                                          ---*
 *---                                                                  ---*
 *---     ----    ----    ----    ----    ----    ----    ----    ----  ---*
 *---                                                                  ---*
 *---     Version 1a              2017 October 25           Joseph Phillips          ---*
 *---                                                                  ---*
 *-------------------------------------------------------------------------*/


#include        <stdlib.h>
#include        <stdio.h>
#include        <string.h>


//  PURPOSE:  To distinguish among the different ways something can be washed.
typedef enum
      {
        NO_WASH,
        HAND_WASH,
        MACHINE_WASH
      }
      howWash_ty;


//  PURPOSE:  To distinguish among the temperatures at which something is
//        washed.
typedef enum
      {
        NO_TEMP,
        COLD_TEMP,
        WARM_TEMP,
        HOT_TEMP
      }
      temperature_ty;


//  PURPOSE:  To distinguish among the different ways something can be dried.
typedef enum
    {
       NO_DRY,
       LINE_DRY,
       TUMBLE_DRY
      }
      howDry_ty;


//  PURPOSE:  To distinguish among the things that can be washed.
typedef      enum
      {
       LIGHTS_WASHED,
       DARKS_WASHED,
       DELICATES_WASHED,
       TROUSERS_WASHED,
       SHIRTS_WASHED,
       UNDERWEAR_WASHED,
       SHEETS_WASHED
      }
      washWhat_ty;


//  PURPOSE:  To keep track of what is washed and dried, and how they are
//      washed and dried.
struct LaundrySummary
{
  howWash_ty        howWash_;
  temperature_ty    temp_;
  howDry_ty         howDry_;
  int               washWhatBitVector_;


  //  PURPOSE:  To initialize '*this' to its default values.  No parameters
  //      No return value.
  LaundrySummary    ()
  {
    howWash_        = NO_WASH;
    temp_                 = NO_TEMP;
    howDry_               = NO_DRY;
    washWhatBitVector_        = 0;
  }


  //  PURPOSE:  To set '*this' equal to 'source'.  Returns reference to
  //        '*this'.
  LaundrySummary&
                  operator=       (const LaundrySummary&     source
                                  )
  {
    if  (this != &source)
    {
      howWash_                    = source.howWash_;
      temp_                 = source.temp_;
      howDry_               = source.howDry_;
      washWhatBitVector_= source.washWhatBitVector_;
    }

    return(*this);
  }


  //  PURPOSE: To erase everything in '*this', like a default constructor.
  //       No parameters.  No return value.
  void            clear           ()
  {
    temp_                 = NO_TEMP;
    howWash_              = NO_WASH;
    howDry_               = NO_DRY;
    washWhatBitVector_        = 0;
  }


  //  PURPOSE:  To note that 'washWhat' was processed.  No return value.
  void            record          (washWhat_ty washWhat
                                  )
  {
    washWhatBitVector_        |= (1 << washWhat);
  }
```

```
  //  PURPOSE:  To return 'true' if 'washWhat' was laundered, or 'false'
  //       otherwise.
  bool              isMentioned    (washWhat_ty washWhat
                                   )
                                   const
  {
    return( (washWhatBitVector_ & (1 << washWhat)) != 0 );
  }

};


//  PURPOSE:  To note that the parser manipulates 'LaundrySummary' instances.
#define                 YYSTYPE               struct LaundrySummary


const int         LINE_LEN      = 256;


//  PURPOSE:  To point to the current position at which tokenizing should be
//         done.
extern
char*             textPtr;

//  PURPOSE:  To point to the end of the tokenizing input.
extern
char*             textEndPtr;

//  PURPOSE:  To hold the result computed by the parser
extern
LaundrySummary      result;


//  PURPOSE:  To print parse-time error 'cPtr'.  No return value.
extern
int               yyerror          (const char*
                                   );

extern
int               yylex            ();
```

6. **laundryLang.lex**

7. %{

```
/*----------------------------------------------------------------------*
 *---                                                              ---*
 *---            laundryLang.lex                                   ---*
 *---                                                              ---*
 *---      This file defines a tokenizer for the laundry language.  ---*
 *---                                                              ---*
 *---   ----   ----   ----   ----   ----   ----   ----   ----      ---*
 *---                                                              ---*
 *---   Version 1a         2017 October 25        Joseph Phillips   ---*
 *---                                                              ---*
 *----------------------------------------------------------------------*/


//  laundryLang.lex
//  unix> flex -o laundryLang.c laundryLang.lex
//  unix> gcc laundryLang.c -c
//  unix> gcc -o laundryLang laundryLang.tab.o laundryLang.o

#include        "laundryLang.h"
#include        "laundryLang.tab.h"

#undef          YY_INPUT
#define                 YY_INPUT(buffer,result,maxSize)             \
                { result = ourInput(buffer,maxSize); }

extern
int             ourInput(char* buffer, int maxSize);

#define                 MIN(x,y)        (((x)<(y)) ? (x) : (y))

%}

%%


%%

//  PURPOSE:  To return the next char of input.
int             ourInput(char* buffer, int maxSize)
{
  int     n       = MIN(maxSize,textEndPtr - textPtr);
  if  (n > 0)
  {
    memcpy(buffer,textPtr,n);
    textPtr         += n;
  }

  return(n);
}

int                  yywrap()        { return(1); }
```

8. **laundryLang.y**
9. %{

```
/*-------------------------------------------------------------------------*
 *---                                                                   ---*
 *---              laundryLang.y                                        ---*
 *---                                                                   ---*
 *---        This file defines a grammar for the laundry language.     ---*
 *---                                                                   ---*
 *---     ----     ----     ----     ----     ----     ----     ----     ----     ---*
 *---                                                                   ---*
 *---     Version 1a          2017 October 25          Joseph Phillips          ---*
 *---                                                                   ---*
 *-------------------------------------------------------------------------*/


//  $ bison --verbose -d --debug laundryLang.y
//  $ gcc laundryLang.tab.c -c

#include  "laundryLang.h"

%}



%%



%%
```

```
//  PURPOSE:  To name the values of 'washWhat_ty'.
const char* washWhatNameArray[]   = {
                                      "lights",
                                      "darks",
                                      "delicates",
                                      "trousers",
                                      "shirts",
                                      "underwear",
                                      "sheets"
                                    };


//  PURPOSE:  To point to the current position at which tokenizing should be
//        done.
char*           textPtr         = NULL;

//  PURPOSE:  To point to the end of the tokenizing input.
char*           textEndPtr      = NULL;

//  PURPOSE:  To hold the result computed by the parser
LaundrySummary      result;



//  PURPOSE:  To print the result of the laundry order 'toPrint'.  No return
//        value.
void            print           (const LaundrySummary&     toPrint
                                )
{

  for  (int i = (int)LIGHTS_WASHED;  i <= (int)SHEETS_WASHED;  i++)
  {
    washWhat_ty         washWhat       = (washWhat_ty)i;

    if  ( !toPrint.isMentioned(washWhat) )
    {
      continue;
    }

    switch  (toPrint.howWash_)
    {
    case NO_WASH :
```

```
  continue;

case HAND_WASH :
  printf("Hand wash ");
  break;

case MACHINE_WASH :
  printf("Machine wash ");
  break;
}

printf("%s ",washWhatNameArray[washWhat]);

switch  (toPrint.temp_)
{
case NO_TEMP :
  break;

case COLD_TEMP :
  printf("in cold water.");
  break;

case WARM_TEMP :
  printf("in warm water.");
  break;

case HOT_TEMP :
  printf("in hot water.");
  break;
}

switch  (toPrint.howDry_)
{
case NO_DRY :
  break;

case LINE_DRY :
  printf("  Line dry.");
  break;

case TUMBLE_DRY :
  printf("  Tumble dry.");
  break;
```

```
      }

      printf("\n");
    }
  }


  //  PURPOSE:  To print parse-time error 'cPtr'.  No return value.
  int        yyerror (const char *cPtr)
  {
    printf("%s, sorry!\n",cPtr);
    return(0);
  }


  //  PURPOSE:  To get input, run the parser, and display the result if the
  //         parse was successful.
  int        main    (int argc, char* argv[])
  {
    char   line[LINE_LEN];

    if  (argc >= 2)
      textPtr        = argv[1];
    else
    {
      printf("Please enter an expression: ");
      textPtr = fgets(line,LINE_LEN,stdin);
    }

    textEndPtr     = textPtr + strlen(textPtr);

    if  (yyparse() == 0)
    {
      print(result);
    }

    return(EXIT_SUCCESS);
  }
```

10. **Please define the tokens towards the top of laundryLang.y:**
11. You have to define the starting non-terminal with %start, and all the tokens with %token. Do not forget the period (.), it is part of the grammar.
12. **Please define the regular expressions in laundryLang.lex:**

13. These are very straight-forward. It would be nice to recognize both capitalized and lowercase forms of "machine", "hand", "tumble" and "line".
14. **Please define the grammar rules in the middle of laundryLang.y:**
15. Be sure to:
    - Remember, the underlying type that you are manipulating (the YYSTYPE) is struct LaundrySummary. Please look at its fields in laundryLang.h.
    - Whenever you use a particular struct LaundrySummary instance for the first time, it is best to run the clear() method on it.
    - Then, run either the record() method, or set its howWash_, temp_ or howDry_ member variable appropriately. For example, this is my code for handling the rule type -> LIGHTS
    - $$.clear();
      $$.record(LIGHTS_WASHED);

    - Rules like wash -> MACHINE WASH temp what will require that you combine data from several right-hand side struct LaundrySummary instances.
    - The rule itemList -> item itemList will require that you combine the washWhatBitVector_ bit field.
    - In the final rule for s set the global variable result equal to the resulting struct LaundrySummary instance.