```
Ray Mitchell, U99999999
MeanOldTeacher@MeanOldTeacher.com
C/C++ Programming II
Section 149123, Ray Mitchell
June 25, 2019
C2A3E1_Sentences.txt
Right-Left Rule Sentences

 1.  fish decays to a pointer to a double.
 2.  fish decays to a pointer to a double.
 3.  fish decays to a pointer to a double.
 4.  fish is an array of 57 doubles.
 5.  fish decays to a pointer to a double.
 6.  fish is an array of 57 doubles.
 7.  fish is an array of 57 doubles.
 8.  fish decays to a pointer to a double.
 9.  fish decays to a pointer to a double.
10.  fish decays to a pointer to a double.
```

```
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming II
// Section 149123, Ray Mitchell
// June 25, 2019
// C2A3E2_TestDeclarations.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function TestDeclarations, which implements various
// declarations and a cast.
//

const int LENGTH = 6;       // number of elements in each array

//
// Demonstrates various declarations and a cast, including the initialization
// of two of the variables.
//
void TestDeclarations()
{
   long **(*afe)[LENGTH];               // 1.
   float (*pv)(int (*pa)[LENGTH]) = 0; // 2.
   afe = (long **(*)[LENGTH])pv;       // 3.
   int &rF1(double *precision);        // 4.
   int *rF3(double &precision);        // 5.
}
```

```c
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming II
// Section 149123, Ray Mitchell
// June 25, 2019
// C2A3E3_RecordOpinions.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function RecordOpinions, which prompts the user to input
// survey values then displays a table of the results.
//

#include <stdio.h>

#define ENDPOINT 5                          // abs(lowest/highest) response
#define BEST ENDPOINT                       // highest response value
#define WORST (-ENDPOINT)                   // lowest response value
#define RESPONSES (2 * ENDPOINT + 1)        // # of different response values
#define TERMINATE 999                       // termination code

//
// Tally user responses to prompts for numeric values and display a count of the
// number of users giving each response value.  Response values in the range
// -ENDPOINT <= response <= ENDPOINT are used as direct indices into the array.
// When the user enters the termination value in <TERMINATE> or an illegal
// character the algorithm stops gathering user input and outputs the results.
//
void RecordOpinions(void)
{
   int responseArray[RESPONSES] = {0};                // holds responses
   int *resPtr = &responseArray[ENDPOINT];            // array midpoint
   int response;

   do
   {
      //
      // Get a user response, check its validity, & update response count if the
      // response is in range.
      //
      printf("Enter one of [%d,%d], or %d to end: ", WORST, BEST, TERMINATE);

      // If illegal character terminate input to prevent infinite loop...
      if (scanf("%d", &response) != 1)
      {
         fprintf(stderr, "  Illegal input character; survey terminated\n");
         response = TERMINATE;
      }
      // else, if user entered termination value...
      else if (response == TERMINATE)
         printf("  Survey terminated by user\n");
      // else, if user entered out of range value...
      else if (response < WORST || response > BEST)
         fprintf(stderr, "  Out of range input rejected: %d\n", response);
      // else, entry was acceptable; update response count.
      else
      {
         ++resPtr[response];
         printf("  Input accepted: %d\n", response);
      }
   } while (response != TERMINATE);

   // For each rating, display the number of respondents...
   printf("\n\nRating       Responses\n"           // print resp...
             "------       ---------\n");         // ...table header
   for (response = WORST; response <= BEST; ++response)
      printf("%4d%14d\n", response, resPtr[response]);
}
```

```c
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming II
// Section 149123, Ray Mitchell
// June 25, 2019
// C2A3E4_OpenFile.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function OpenFile, which opens the file specified by its
// parameter in the read-only mode.
//

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

//
// Open the file named in <fileName> and return its FILE pointer if the open
// succeeds.  If it fails display an error message and terminate the program
// with an error code.
//
FILE *OpenFile(const char *fileName)
{
   // Open the file in the read-only mode & check for failure.
   FILE *fp;
   if ((fp = fopen(fileName, "r")) == NULL)
   {
      // Display an error message and terminate with an error exit code.
      fprintf(stderr, "File \"%s\" didn't open.\n", fileName);
      exit(EXIT_FAILURE);
   }
   return fp;
}
```

```c
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming II
// Section 149123, Ray Mitchell
// June 25, 2019
// C2A3E4_ParseStringFields.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function ParseStringFields, which extracts and displays
// substrings from lines in the open text file specified by its parameter.
//

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

#define MAXLINE 256                  // size of temporary input buffer
#define DELIMITERS "aeiouAEIOU\t\n" // token delimiters

//
// Parse the text in file <fp> and break it into tokens separated by the
// delimiters specified by <DELIMITERS>.  Display each token on a separate
// line, omitting any leading whitespace in the token.
//
void ParseStringFields(FILE *fp)
{
   // Get successive lines of text from the open file in <fp>.
   char buf[MAXLINE];
   while (fgets(buf, (int)sizeof(buf), fp) != NULL)
   {
      // Break the line of text into separate tokens.
      for (char *chPtr = buf; chPtr = strtok(chPtr, DELIMITERS); chPtr = NULL)
      {
         // Skip leading whitespace in the current token.
         while (isspace(*chPtr))
            ++chPtr;
         // Display what remains of the token on its own line.
         puts(chPtr);
      }
   }
}
```