Exercise 1 *(2 points – Short answers only – No program required)*

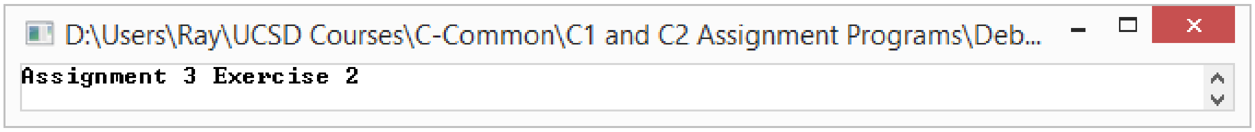
The actual data type of identifier **values** is "array of 25 **float**s". However, depending upon the context in which it appears it will either be treated as an "array of 25 **floats**" (the non-decaying case) or a "pointer to a **float**" (the decaying case). It is always wrong to simply state that the data type of **values** is an "array of **float**s", since that would imply that an "array of 25 **float**s" and an array of any other number of **float**s" are the same data type.

1. **values** decays to a pointer to a **float**
2. **values** is an array of 25 **float**s
3. **values** decays to a pointer to a **float**
4. **values** decays to a pointer to a **float**
5. **values** decays to a pointer to a **float**
6. **values** decays to a pointer to a **float**
7. **values** is an array of 25 **float**s
8. **values** decays to a pointer to a **float**
9. **values** decays to a pointer to a **float**
10. **values** decays to a pointer to a **float**

Exercise 2 *(4 points – C++ Program)*

```
//
// ...the usual title block Student/Course/Assignment/Compiler information goes here...
//
// This file contains function TestDeclarations, which merely implements
// some instructor-specified declarations and typecasts.
//

const int ARRAY_SIZE = 9;        // number of elements in each array

//
// Demonstrate various declarations and a typecast, including the
// initialization of three of the variables.
//
void TestDeclarations()
{
   void *vp = 0;                      // 1.
   int fcnA(int val);                 // 2.
   float (**ppa)[ARRAY_SIZE];         // 3.
   float (**&rppa)[ARRAY_SIZE] = ppa;  // 4.
   ppa = (float (**)[ARRAY_SIZE])vp;   // 5.
}
```

C2A3E2 Screen Shot


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

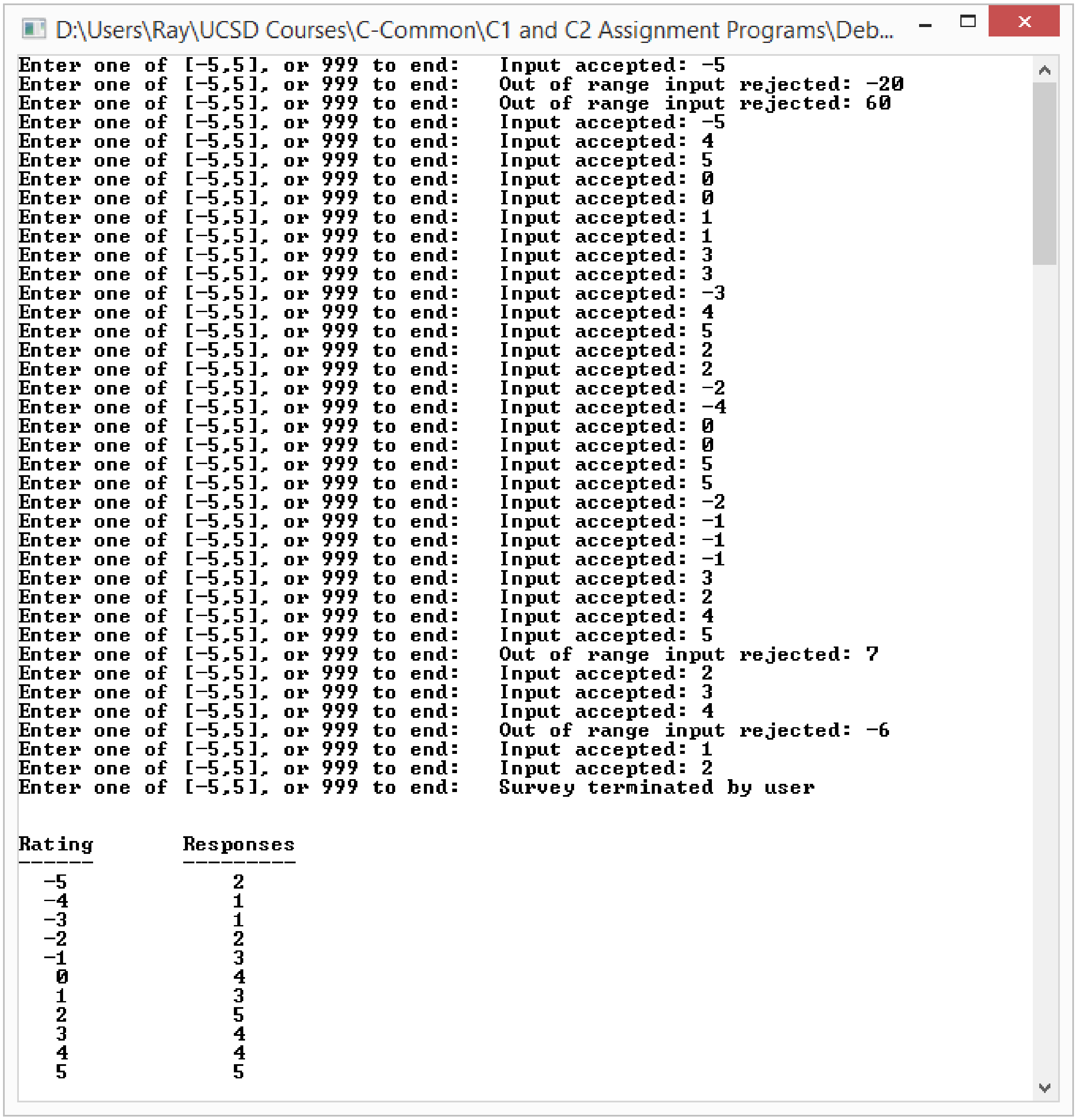
```
Assignment 3 Exercise 2
```

Exercise 3 *(6 points – C Program)*

```
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * 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);
      }
```

```c
   } 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]);
}
```

C2A3E3 Screen Shot


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

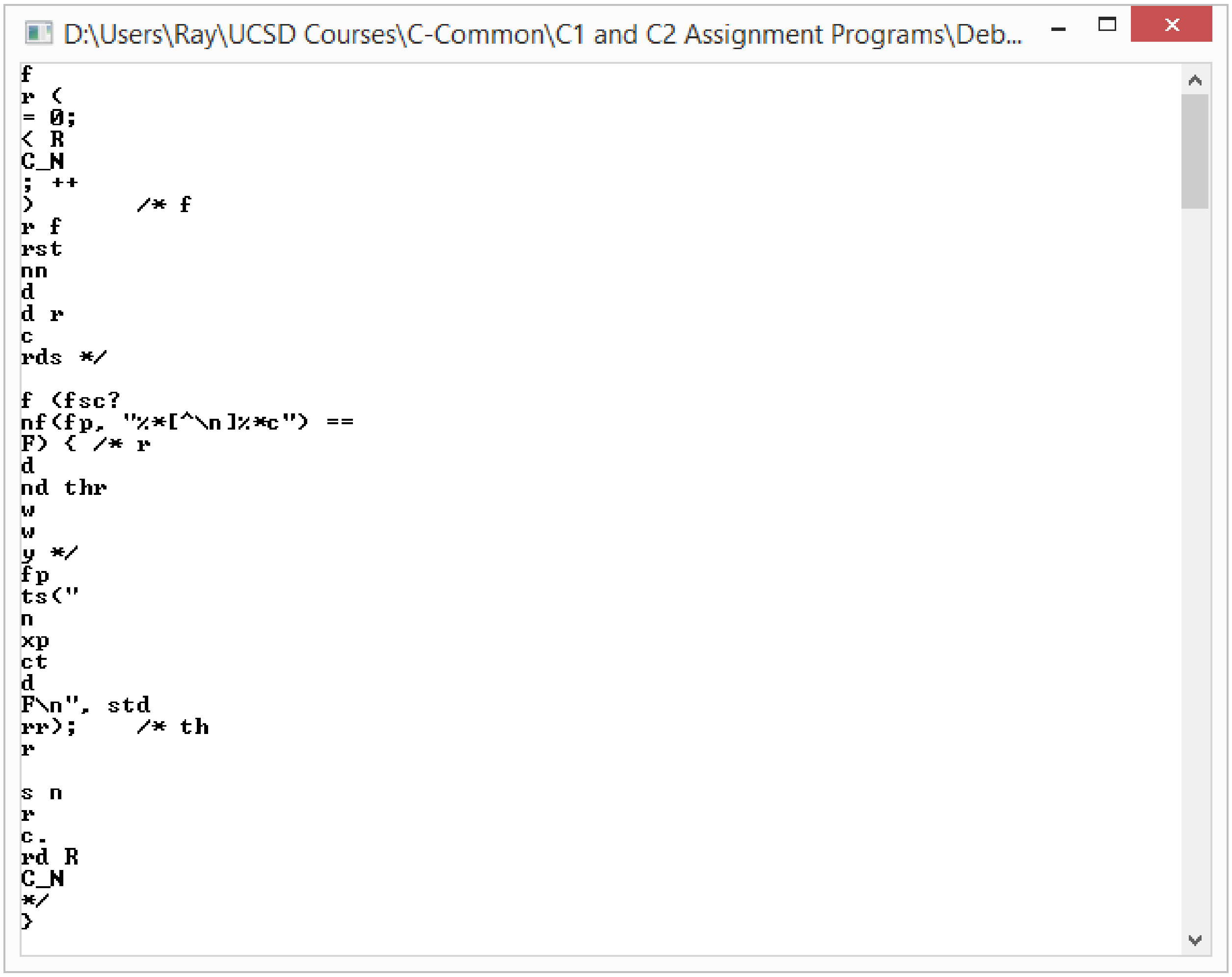
```
Enter one of [-5,5], or 999 to end:   Input accepted: -5
Enter one of [-5,5], or 999 to end:   Out of range input rejected: -20
Enter one of [-5,5], or 999 to end:   Out of range input rejected: 60
Enter one of [-5,5], or 999 to end:   Input accepted: -5
Enter one of [-5,5], or 999 to end:   Input accepted: 4
Enter one of [-5,5], or 999 to end:   Input accepted: 5
Enter one of [-5,5], or 999 to end:   Input accepted: 0
Enter one of [-5,5], or 999 to end:   Input accepted: 0
Enter one of [-5,5], or 999 to end:   Input accepted: 1
Enter one of [-5,5], or 999 to end:   Input accepted: 1
Enter one of [-5,5], or 999 to end:   Input accepted: 3
Enter one of [-5,5], or 999 to end:   Input accepted: 3
Enter one of [-5,5], or 999 to end:   Input accepted: -3
Enter one of [-5,5], or 999 to end:   Input accepted: 4
Enter one of [-5,5], or 999 to end:   Input accepted: 5
Enter one of [-5,5], or 999 to end:   Input accepted: 2
Enter one of [-5,5], or 999 to end:   Input accepted: 2
Enter one of [-5,5], or 999 to end:   Input accepted: -2
Enter one of [-5,5], or 999 to end:   Input accepted: -4
Enter one of [-5,5], or 999 to end:   Input accepted: 0
Enter one of [-5,5], or 999 to end:   Input accepted: 0
Enter one of [-5,5], or 999 to end:   Input accepted: 5
Enter one of [-5,5], or 999 to end:   Input accepted: 5
Enter one of [-5,5], or 999 to end:   Input accepted: -2
Enter one of [-5,5], or 999 to end:   Input accepted: -1
Enter one of [-5,5], or 999 to end:   Input accepted: -1
Enter one of [-5,5], or 999 to end:   Input accepted: -1
Enter one of [-5,5], or 999 to end:   Input accepted: 3
Enter one of [-5,5], or 999 to end:   Input accepted: 2
Enter one of [-5,5], or 999 to end:   Input accepted: 4
Enter one of [-5,5], or 999 to end:   Input accepted: 5
Enter one of [-5,5], or 999 to end:   Out of range input rejected: 7
Enter one of [-5,5], or 999 to end:   Input accepted: 2
Enter one of [-5,5], or 999 to end:   Input accepted: 3
Enter one of [-5,5], or 999 to end:   Input accepted: 4
Enter one of [-5,5], or 999 to end:   Out of range input rejected: -6
Enter one of [-5,5], or 999 to end:   Input accepted: 1
Enter one of [-5,5], or 999 to end:   Input accepted: 2
Enter one of [-5,5], or 999 to end:   Survey terminated by user


Rating       Responses
------       ---------
  -5             2
  -4             1
  -3             1
  -2             2
  -1             3
   0             4
   1             3
   2             5
   3             4
   4             4
   5             5
```

Exercise 4 *(8 points – C Program)*

```
    **************************************** FILE C2A3E4_OpenFile.c ****************************************
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * 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)
{
   FILE *fp;
   /* Open the file in the read-only mode & check for failure. */
   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;
}
```

```
   ************************************* FILE C2A3E4_ParseStringFields.c *************************************
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * 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\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)
   {
        char *chPtr;
        /* Break the line of text into separate tokens. */
        for (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);
        }
     }
  }
```

C2A3E4 Screen Shot


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
f
r (
= 0;
< R
C_N
; ++
)       /* f
r f
rst
nn
d
d r
c
rds */

f (fsc?
nf(fp, "%*[^\n]%*c") ==
F) { /* r
d
nd thr
w
w
y */
fp
ts("
n
xp
ct
d
F\n", std
rr);    /* th
r

s n
r
c.
rd R
C_N
*/
}
```