Solutions to Chapter 11

Review Questions

1. b. False
3. b. False
5. e. string
7. a. The assignment operator copies the value of a string variable to another.
9. c. *gets*
11. c. `if (strmcp (string1, string2) == 0)`
13. a. *strchr*
15. e.functional

Exercises

17.

```
*(y)     ==> e
*(y + 1) ==> a space (between e and i)
*(y + 4) ==> a space (between s and b)
```

19.

A number > 0 (Probably 57, which is the difference between `'z'` (122) and `'A'` (65))

21.

```
abefgnpanm
panm
befgnpanm
```

There is nothing to be printed for the fourth line because `s9` is pointing null.

23.

```
DOO
CK
BOO
O
OO
OBOO
B
```

25.

```
argorp
agrxxx
```

Note: `xxx` on the second line would be anything that exists in memory because `p` is pointing before the address of the beginning of the string.

Problems

27. See Program 11-1.

Program 11-1 Solution to Problem 27

```
/* ================ delFirst ====================
   This function deletes the first character of the
   string received from the main.
      Pre x   is a pointer to a non-null string
      Post    first character deleted
*/
void delFirst (char* x)
{
// Local Declarations
   char* walker;

// Statements
   walker = x + 1;
   while (*walker != '\0')
      {
       *(walker - 1) = *walker;
       walker++;
      } // while
   *(walker - 1) = *walker;
   return;
}  // delFirst
```

29. See Program 11-2.

Program 11-2 Solution to Problem 29

```
 /*================ delSpace ==================
   This function deletes leading spaces from a string
   received from main.
      Pre   str is a pointer to the string.
      Post  Leading spaces deleted
*/
void delSpace (char* str)
{
// Local Declarations
   char* pData;
   char* pFront;

// Statements
   pData = str;
   while (isspace(*(pData)))
      pData++;

   pFront = str;
   while (*pData != '\0')
      *(pFront++) = *(pData++);
   *pFront = '\0';
   strcpy (pData, pFront);
   return;
}  // delSpace
```

31. See Program 11-3.

Program 11-3 Solution to Problem 31

```
/* ================= insertString ====================
   This function inserts a string into another string
   at specified position.
      Pre   str1 the receiving string
```

Program 11-3 Solution to Problem 31 (continued)

```
            str2 the string to be inserted
            insertion (index) position in first string
              (combined length of strings must be < 80)
      Post  successful, returns positive number
            !successful, return zero
*/
int insertString (char* str1, char* str2, int idx)
{
// Local Declarations
   char* ptr;
   char  temp[81];
   int   retval = 1;

// Statements
   if (idx > strlen (str1) || idx < 0)
       retval = 0;
   else
      {
       ptr = str1 + idx;
       strcpy (temp, str2);
       strcat (temp, ptr);
       *ptr = '\0';
       strcat (str1, temp);
      } // else
   return retval;
}  // insertString
```

33. See Program 11-4.

Program 11-4 Solution to Problem 33

```
/* ================== newStrCpy =================
   This function does the same job as strcpy.
      Pre   dest is the destination string
            src is the source string
      Post  string copied
*/
char* newStrCpy (char* dest, const char* src)
{
// Local Declarations
   char* toPtr;

// Statements
   for (char* fromPtr = src, toPtr = dest;
             *fromPtr != '\0';
              fromPtr++, toPtr++)
      *toPtr = *fromPtr;

   *toPtr = *fromPtr;
   return dest;
}  // newStrCpy
```

35. See Program 11-5.

Program 11-5 Solution to Problem 35

```
/* ================== newStrCmp ==============
   This function does the same job as strcmp.
      Pre   s1 and s2 are the strings to compare
      Post  return result of comparing
*/
int newStrCmp (const char* s1, const char* s2)
{
// Statements
```

Program 11-5 Solution to Problem 35 (continued)

```
   while (*s1 == *s2 && *s1 != '\0' && *s2 != '\0')
      {
       s1++;
       s2++;
       } // while

   return (*s1 - *s2);
}  // newStrCmp
```

37. See Program 11-6.

Program 11-6 Solution to Problem 37

```
/* This program reads a text file and removes any extra
   spaces after a period, comma, semicolon, or colon
   and writes the corrected text to a new file.
      Written by:
      Date:
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define IN_FILE  "FILE1.TXT"
#define OUT_FILE "FILE2.TXT"
#define SPACE    " "
#define TOKENS   ".,:;"

int main (void)
{
// Local Declarations
   FILE* fpTextIn;
   FILE* fpTextOut;
   char  strIn [256];
   char* ptrIn;

// Statements
   printf ("*** Start Program \n\n");

   if (!(fpTextIn = fopen(IN_FILE, "r")))
      printf("\aERROR: Cannot open %s\n", IN_FILE),
          exit (100);
   if (!(fpTextOut = fopen(OUT_FILE, "w")))
      printf("\aERROR: Cannot open %s\n", OUT_FILE),
          exit (200);

   while (fgets (strIn, 255, fpTextIn))
      {
       *(strIn + 255) = '\0';

       ptrIn = strtok (strIn, SPACE);
       fputs (ptrIn, fpTextOut);
       ptrIn = strtok (NULL, SPACE);

       while (ptrIn)
          {
           fputc (' ', fpTextOut);
           if (strchr (TOKENS, *(ptrIn - 2)))
              {
               while (*ptrIn == ' ')
                  ptrIn++;
              } // if last char before space is .,:;
           fputs (ptrIn, fpTextOut);
           ptrIn = strtok (NULL, SPACE);
          } // while more tokens
      } // while !EOF
```

Program 11-6 Solution to Problem 37 (continued)

```
    fclose (fpTextIn);
    fclose (fpTextOut);
    printf ("\n\n*** End Program ***\n");
    return 0;
}   // main
```