Solutions to Chapter 6

Review Questions

1. a. True
3. b. False
5. e. The update for a pretest loop must be a part of the loop body.
7. e. The number of updates always equals the number of loop iterations.
9. e. Both the *for* and the *while*.
11. d. Both statements include initialization within the statements.
13. e. May be linear logarithmic or quadratic

Exercises

15.

a. prints “12” infinite number of times on separate lines

b. prints "12" infinite number of times on separate lines

c. prints “12” infinite number of times on separate lines

17.

a. On separate lines

```
12      10      8
```

b. On separate lines

```
12      10      8
```

19.

a.

```
for (x = 0; x < 10; x++)
      printf ("%d\n", x);
```

b.

```
for (scanf ("%d", &x); x != 9999; scanf ("%d", &x))
      printf ("%d\n", x);
```

21.

a.

```
x = 1;
while (x < 100)
     {
      printf ("%d\n", x);
      x++;
     } // while
```

b.

```
 while (scanf ("%d", &x) != EOF)
        printf("%d\n", x);
```

23.

a.

```
x = 0;
printf ("%d\n", x);
x++;
while (x < 100)
   {
    printf ("%d\n", x);
    x++;
   } // while
```

b.

```
res = scanf ("%d", &x);
while (res != EOF)
   res = scanf ("%d", &x);
```

25. The *for* loop prints the numbers 0 to 9 all on the same line with no spaces. The following would be one way to correct the code:

```
for (num = 1; num <= 10; num++)
         printf (" %d ", num);
```

27.

a. On separate lines:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

b. On separate lines:

```
1 3 5 7 9 11 13 15 17 19
```

29.

a. The numbers 1 to 20 would each be printed 5 times on separate lines:

```
11111
22222
  .
  .
  .
2020202020
```

b. This loop prints the number 20 twenty times on a line, then prints the number 19 nineteen times on a line, then prints the number 18 eighteen times on a line, and so on, until it prints the number 1 once.

31.

a. Since the for loop has no limit test expression and no update expression, the loop will run forever.

b. Since there is no limit condition, the loop can only be exited by using one of the jump statements: *break*, *return* or *goto*.

c. This is not good structured programming style. Good programming style follows the accepted conventions. Programmers expect to find the expressions associated with the for loop in their proper places within the *for* statement. To put them elsewhere in the loop body is not what is expected and therefore tends to confuse the reader. Additionally, these statements are much more error prone when placed in the body of the loop, especially if the body has complex selection statements controlling various parts of the code.

Problems

33.

a.

```
for (i = 6; i <= 66; i += 2)
     printf ("%d, ", i);
printf ("\b\b\n");
```

b.

```
for (i = 7; i <= 67; i += 2)
     printf ("%d, ", i);
printf ("\b\b\n");
```

c.

```
for (sum = 0,  i = 1; i <= 15; i++)
     sum += i;
```

d.

```
for (sum = 0, i = 15; i <= 45; i += 2)
     sum += i;
```

e.

```
for (sum = 0, i = 0; i < 50; i++)
      {
       printf ("%d + ", i * 3 + 1);
       sum += (i * 3 + 1);
      } // for
   printf ("\b\b\b = %d", sum);
```

35. See Program 6-1.

Program 6-1 Solution to Problem 35

```
/* Calculate average of 'n' negative floating point
   numbers.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int   n;
   int   count;
   float num;
   float sum;

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

   printf ("Please enter number of entries to use:  ");
   scanf  ("%d", &n);

   count = 0;
   sum   = 0;


   for (int i = 0; i < n; i++)
        {
         printf ("Please enter a number:  ");
         scanf  ("%f", &num);
         if (num < 0)
            {
             sum += num;
             count++;
```

Program 6-1 Solution to Problem 35 (continued)

```
            } // if
         } // for
   printf ("\nThe average is: %f.\n", sum / count);
   printf ("\n***  End of Program ***\n");

   return 0;
}  // main
```

37. See Program 6-2.

Program 6-2 Solution to Problem 37

```
/* Create a pattern of sequential numbers.
     Written by:
     Date:
*/
#include <stdio.h>

int main (void)
{
// Statements
   printf ("*** Start of Program ***\n\n");

   for (int i = 9; i >= 1; i--)
       {
        for (int j = 1; j <= i; j++)
             printf ("%d ", j);
        printf ("\n");
       } // for
   printf ("\n\n*** End of Program ***\n");
   return 0;
}  // main
```

39. See Program 6-3.

Program 6-3 Solution to Problem 39

```
/* ================= pattern ================
   Create a pattern given a height and width.
      Pre   Given height and width
      Post  Pattern printed
*/
void pattern (int height, int width)
{
// Statements
   for (int i = 1; i <= height; i++)
       {
        if (i == 1 || i == height)
            for (int j = 1; j <= width; j++)
                 printf ("=");
        else
           {
            for (int j = 1; j <= width; j++)
                {
                 if (j == 1 || j == width)
                     printf ("*");
                 else
                     printf (" ");
                } // for
           } // else

        printf ("\n");
       } // for
```

Program 6-3 Solution to Problem 39 (continued)

```
   return;
}  // pattern
```

41. See Program 6-4.

Program 6-4 Solution to Problem 41

```
/* ==================== pattern ====================
   This function creates a pattern given the height.
      Pre   Given height
      Post  Pattern printed
*/
void pattern (int height)
{
// Statements
   for (int i = height - 1; i >= 0; i--)
       {
        for (int j = 1; j <= i * 2 + 1; j++)
             printf ("*");
        printf ("\n");
       } // for

   return;
}  // pattern
```

43. See Program 6-5.

Program 6-5 Solution to Problem 43

```
/* This program modifies Program 6-3 to
   display the total as each number is entered.
      Written by:
      Date:
*/
#include <stdio.h>

int main (void)
{
// Local Definitions
   int num;
   int sum = 0;

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

   printf ("\nEnter your numbers: <EOF> to stop.\n");
   while  (scanf ("%d", &num) != EOF)
          {
           sum = sum + num;
           printf ("Total: %8d\n", sum);
          } // while
   printf ("\n\n*** End of Program ***\n");

   return 0;
}  // main
```

45. See Program 6-6.

Program 6-6 Solution to Problem 45

```
/* Test function that reads only positive numbers.
      Written by:
      Date:
```

Program 6-6 Solution to Problem 45 (continued)

```
*/
#include <stdio.h>
#include <stdbool.h>

// Function Declaration
int getValid (void);

int main (void)
{
// Local Declarations
   int num;

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

   while ((num = getValid ()) != EOF)
       printf ("Valid number: %8d\n", num);

   printf ("\n\n*** End of Program ***\n");
   return 0;
}  // main

/* ====================== getValid =======================
   Verify that the input is a positive, even number.
      Pre   nothing
      Post  Reads and returns a valid number or EOF
*/
int getValid (void)
{
// Local Definitions
   int  num;
   bool valid = false;

// Statements
   do
      {
       printf ("\nEnter an integer or <EOF> to stop: ");
       scanf ("%d", &num);

       //Validate input
       if (feof(stdin))
          {
           valid = true;
           num   = EOF;
          } // if
       else if (num < 0)
           printf ("\n\a Number must be positive.\n");
       else if (num % 2)
           printf ("\n\aNumber must be even.\n");
       else
           valid = true;
      } while (!valid);
   return num;
}  // getValid
```

47. See Program 6-7.

Program 6-7 Solution to Problem 47

```
/* ==================== sumEOF2 ====================
   Read a series of numbers, terminated by EOF, and
   return their sum.
      Pre   nothing
      Post  reads data and returns the sum of inputs.
*/
int sumEOF2 (void)
```

Program 6-7 Solution to Problem 47 (continued)

```
{
// Local Declarations
   int num;
   int sum  = 0;
   int done = 0;

// Statements
   printf ("Enter a number (EOF to quit):  ");
   do
      {
       if (scanf  ("%d", &num) == EOF)
          done = 1;
       else
          {
           sum += num;
           printf ("Next number (EOF to quit):  ");
          } // else
      } while (!done);
   return sum;
}  // sumEOF2
```

49. See Program 6-8.

Program 6-8 Solution to Problem 49

```
/* This program approximates Euler's number e using a
   loop that terminates when the difference between two
   successive values differs by less than 0.0000001.
      Written by:
      Date:
*/
#include <stdio.h>

// Prototype Declarations
   long factorial (long);

int main (void)
{
// Local Declarations
   int    n      =  0;
   double old_e  =  0.0;
   double new_e  =  1.0;

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

   while (new_e - old_e > .0000001)
         {
          old_e  =  new_e;
          new_e +=  1.0 / factorial (++n);
         } // while

   printf ("Successive 'e's are %10.8lf & %10.8lf\n",
            old_e, new_e);
   printf (" (n = %d and %d)\n", n - 1 , n);
   printf ("\nThe difference is %10.8lf\n",
            new_e - old_e);

   printf ("\n\n*** End of Program ***\n");
   return 0;
}  // main

/* ==================== factorial ====================
   Calculate factorial of a number using recursion.
   There is no test that the result fits in a long.
      Pre  n is the number being raised factorially
```

Program 6-8 Solution to Problem 49 (continued)

```
      Post result is returned
*/
long factorial (long n)
   {
   // Statements
   if (n == 0)
      return 1;
   else
      return (n * factorial (n - 1));
}  // factorial
```

51. See Program 6-9.

Program 6-9 Solution to Problem 51

```
/* This program reads an integer from the keyboard and
   then calls a recursive function to print the digits
   in reverse order.
      Written by:
      Date:
*/
#include <stdio.h>

// Function Declarations
   void printReversed (int original);

int main (void)
{
// Local Declarations
   int original;

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

   printf ("\nEnter the number: ");
   scanf  ("%d", &original);

   printf ("\nThe original number was: %d", original);
   printf ("\nThe reversed number is : ");
   printReversed (original);

   printf ("\n\n*** End of Program ***\n");

   return 0;
}  // main

/* ================== printReversed ====================
   This function prints digits in an integer reversed.
      Pre   num contains number to be reversed
      Post  prints number reversed
*/
void printReversed (int num)
{
// Statements
   if (num  !=  0)
      {
       printf ("%d", num % 10);
       printReversed (num / 10);
      } // if num != 0
   return;
}  // printReversed
```