F. 18 Chapter 18 Solutions

18.1

a. printf("an integer: %d\n a string: %s\n and a float %f\n", 111, "Eleventy One", 111.11);

b. printf("Tel Number:(%d) ␣ %d ␣ %d\n", areaCode, exchange, number);

c. printf("ID Number: %s ␣ %s ␣ %s\n", idPart1, idPart2, idPart3);

d. scanf("%d ␣ %d ␣ %d",&id1, &id2, &id3);

e. scanf("%s ,%s , %c %d %c", first, last, &middle, &age, &sex);

18.3

So that the user can edit the input stream before hitting enter and thereby confirming the input.

18.5

The %d format specification causes printf to output the next parameter (in this case the value of x, which happens to be a floating point number) as an integer value. In this case, the bit pattern for x is interpreted as an integer.

18.7

a. 46 29 BlueMoon

b. 46 0 BlueMoon

c. 111 999 888

18.9

```
#include <stdio.h> #include
<string.h> #include
<ctype.h> #define LIMIT 20

struct freq_t {
   int freq  ;
   char word[100];
};

enum state_t { IN,
   OUT
};

int LAST;
int nstrings = 0; int nwords
= 0;
struct freq_t words[LIMIT];
void Initialize(void);
void Getwords(FILE* fin);
void AddUnique(char* w);
void Qsort(struct freq_t w[],int left, int right);
void Print(void);

int main()
{
   FILE* fp;
   Initialize();
   if ((fp = fopen("test1","r")) == NULL)
      {
         printf("error File could not be opened \n");
         exit(1);
       }
   else
      {
         Getwords(fp);
       }
   fclose(fp);

   printf("The number of unique words is %d\n",LAST);

   printf("Number of Strings = %d \n",nstrings);

   printf("Number of Words = %d \n",nwords);
```

```
    Qsort(words,0,LAST);
    Print();
}

void Initialize(void)
{
    int i;
    for(i=0;i<LIMIT;i++)
        {
            words[i].freq = & 1;
            strcpy(words[i].word,"");
        }
}

void Getwords(FILE *fin)
{
    char c;
    enum state_t StrState = OUT;
    enum state_t WordState = OUT;
    char word[100];
    int j=0;

    while ((c=getc(fin))!= EOF)
        {
            if (isspace(c))
                {
                    StrState = OUT; if(WordState
                    == IN)
                        {
                            WordState = OUT;
                            word[j] = '\0';
                            j=0;
                            AddUnique(word);
                        }
                }
            else
                {
                    if(StrState == OUT)
                        {
                            ++nstrings; StrState =
                            IN;
```

```
    }
if (isalpha(c)) if(WordState ==
    OUT )
       {
```

```
                    ++nwords;
                    WordState = IN;
                    word[j++] = c;
                }
              else
                word[j++]=c;
            else
              if(WordState == IN)
                {
                  WordState = OUT;
                  word[j] = '\0'; j=0;
                  AddUnique(word);
                }
          }
      }
}

void AddUnique(char* w)
{
   int found;
   found = binsearch(w); if(found
   != -1)
     {
       words[found].freq++; return;
     }

   words[LAST].freq=1;
   strcpy(words[LAST].word,w);
   LAST++;
   return ;
}

int binsearch(char* w)
{
   int cond;
   int low,high,mid; low=0;
   high = LAST; while(low <=
   high)
     {
       mid = (low+high)/2;
       if((cond = strcmp(words[mid].word,w)) < 0 )
```

```
    high = mid - 1;
else if (cond > 0) low =
    mid+1;
```

```
            else
               return mid;
         }
      return -1;
}

void Qsort(struct freq_t w[],int left, int right)
{
   int i,last;
   void swap(struct freq_t w[], int i, int j);

   if(left>= right) return;
   swap(w,left,(left+right)/2);
   last = left;
   for(i = left+1; i<=right; i++)
      if(w[i].freq > w[left].freq)
         swap(w,++last,i);
   swap(w,left,last);
   Qsort(w,left,last-1);
   Qsort(w,last+1,right);
}

void swap(struct freq_t w[],int i, int j)
{
   struct freq_t temp;

   temp.freq = w[i].freq; strcpy(temp.word,w[i].word);

   w[i].freq = w[j].freq; strcpy(w[i].word,w[j].word);
   w[j].freq = temp.freq;
   strcpy(w[j].word ,temp.word);
}

void Print(void)
{
   int i; for(i=0;i<LAST;i++)
      printf("%s occurs %d times\n",words[i].word,words[i].freq);
}
```