```
01 46 02 6F 01 20 01 62 01 61 01 72 03 21 05 20
```

Run-length encoding works well if the original file contains many long sequences of identical bytes. In the worst case (a file with no repeated bytes), run-length encoding can actually double the length of the file.

(a) Write a program named `compress_file` that uses run-length encoding to compress a file. To run `compress_file`, we'd use a command of the form

`compress_file` *original-file*

`compress_file` will write the compressed version of *original-file* to *original-file*`.rle`.

For example, the command

```
compress_file foo.txt
```

will cause `compress_file` to write a compressed version of `foo.txt` to a file named `foo.txt.rle`. *Hint:* The program described in Programming Project 6 could be useful for debugging.

(b) Write a program named `uncompress_file` that reverses the compression performed by the `compress_file` program. The `uncompress_file` command will have the form

`uncompress_file` *compressed-file*

compressed-file should have the extension `.rle`. For example, the command

```
uncompress_file foo.txt.rle
```

will cause `uncompress_file` to open the file `foo.txt.rle` and write an uncompressed version of its contents to `foo.txt`. `uncompress_file` should display an error message if its command-line argument doesn't end with the `.rle` extension.

8. Modify the `inventory.c` program of Section 16.3 by adding two new operations:

   - Save the database in a specified file.
   - Load the database from a specified file.

   Use the codes `d` (dump) and `r` (restore), respectively, to represent these operations. The interaction with the user should have the following appearance:

```
Enter operation code: d
Enter name of output file: inventory.dat

Enter operation code: r
Enter name of input file: inventory.dat
```

   Hint: Use `fwrite` to write the array containing the parts to a binary file. Use `fread` to restore the array by reading it from a file.

W 9. Write a program that merges two files containing part records stored by the `inventory.c` program (see Programming Project 8). Assume that the records in each file are sorted by part number, and that we want the resulting file to be sorted as well. If both files have a part with the same number, combine the quantities stored in the records. (As a consistency check, have the program compare the part names and print an error message if they don't match.) Have the program obtain the names of the input files and the merged file from the command line.

*10. Modify the `inventory2.c` program of Section 17.5 by adding the `d` (dump) and `r` (restore) operations described in Programming Project 8. Since the part structures aren't stored in an array, the `d` operation can't save them all by a single call of `fwrite`. Instead, it will need to visit each node in the linked list, writing the part number, part name, and quan-