CS 1014: Numerical Computational Techniques

Project 6

Due May 3, 1999

Postal Cost Calculation

All work for this project has to be done by you alone. In case of difficulty, you may seek help from the course instructor or from the course graduate teaching assistant.

Objective:

This project will give you the opportunity to learn how to 1) represent tables using arrays in a Fortran program, 2) declare and initialize arrays in a Fortran program, 3) access array elements, 4) search an array for an item, and 5) declare, implement, and use functions.

Problem Statement:

A small postal system ships packages within your state. Acceptance of parcels is subject to the following constraints:

a. Parcels are not to exceed a weight of 50 pounds.
b. Parcels are not to exceed 3 feet in length, width, or depth.
c. Parcels may not have a combined length and girth exceeding 6 feet.

The girth of a package is the circumference of the package around its smallest sides; the mathematical formula is
*girth = 2 * (s1 + s2 + s3 - largest)*
where *largest* is the largest of the three dimensions, *s1*, *s2*, and *s3*.

Notice that the largest side is indeed the length of the package and accordingly, constraint (c) can be restated as:
(largest + girth) may not exceed 6 feet.

Your program should process a transaction file containing one entry for each box mailed during the week. Each entry contains a *transaction number,* followed by the *weight* of the box and its *dimensions* (the dimensions can be in any order). The program should print the *transaction number, weight,* and *postal charge* for all accepted packages, and the *transaction number* and *weight* for all rejected packages. At the end of the report, the program must print the *number of packages* processed and the *number* rejected.

Initialization of Arrays:

Parcel post table -- weight and cost (contains 25 pairs of values). The parcel post table shown below should be implemented using two one-dimensional arrays in your program. You can determine the postal cost of each parcel by first searching the `weight` array and then using the corresponding element in the `cost` array. For example, if the weight is 10 pounds, then the postal cost will be 5.00. If a package weight falls between weight categories in the table, your program should use the cost for the higher weight. For example, if the weight is 27 pounds (which does not occur in the weight array), then the postal cost will be 9.45.

Parcel Weight	Postal Cost
1	1.50
2	1.75
3	2.00
4	2.50
5	3.00
6	3.50
7	3.75
8	4.00
9	4.40
10	5.00
12	5.45
14	6.00
16	6.50
18	7.00
20	7.75
22	8.25
24	8.65
26	9.15

29	9.45
32	9.85
35	10.40
38	11.00
41	11.80
45	12.50
50	12.95

Input:

Transaction file -- transaction number, weight, and the three dimensions for an arbitrary number of transactions. Assume that all weights are whole numbers, that all dimensions are given to the nearest inch (i.e., all weights and dimensions are integers), and that all data is valid. You may use the following sample data to test your program. The input file must be named as **"PARCEL.DAT"**.

Sample input data:

```
1290    10    10    14    20
1900    20    20    15    10
2090    12    21    25    21
2900    43    39     9     7
2950    40     9    37    12
3000     4     6    12    14
2400    51    10     8     9
3400    23    10     7    20
3450    27     6     9    38
3455    27     6     9    10
2312    18     5     3     9
2410    12     6     9    34
```

Output:

Your program should create and send output to a file named **"REPORT6.OUT"**. Your output should have the exact form shown below. With sample input data, your program should produce output as shown below:

```
Trans. No.          Weight                Postal Charge
-------------------------------------------------------------
1290                10                          5.00
1900                20                          7.75
2090                12          *** unacceptable size or weight ***
2900                43          *** unacceptable size or weight ***
2950                40          *** unacceptable size or weight ***
3000                4                           2.50
2400                51          *** unacceptable size or weight ***
3400                23                          8.65
3450                27          *** unacceptable size or weight ***
3455                27                          9.45
2312                18                          7.00
2410                12                          5.45
-------------------------------------------------------------
Number of packages processed: 12
Number of packages rejected: 5
```

Processing:

You should declare all floating-point variables including the `cost` array as REAL type. You may use the following guidelines to make your program modular.

- You should define and use a function to find the largest (length) of all three sides of a package.
- You should define and use a function to determine whether a package is acceptable or not.
- You should define and use a function to maintain the parcel cost table in two one-dimensional arrays and do the search on the table to return the postal cost of a package from its weight.

You are required to implement and use functions in your program. Points will deducted if you do not use functions.

Documentation:

Do not forget to document your program. Some of your programs will be randomly selected from the archive (maintained on the grader server) later on for manual grading. Points will be deducted if your program is not well documented, well-structured, and readable. You can use the program handed out for the first project as a model for documentation. Do not forget to include your name(s) in the program documentation.

Submission:

You should submit your source code electronically to the automated grader. Do not submit the input or the output file. Before you submit your program, make sure your program runs perfectly and provides right output with the sample input data. As with the fifth programming project, you will be allowed a maximum of four submits. This assignment should be submitted as Project Number 6. **No late submission of this project will be accepted after May 5, 1999.**