Case Study: High Adventure Travel Agency—Part 3

Chapter 6's case study was a program to assist the High Adventure Travel Agency in calculating the costs of their four vacation packages. In Chapter 11, a modification to that program was started, which will be completed here.

Recall that in Chapter 11, data structures were designed to hold all the cost information for a vacation package. You are now ready to add file I/O capabilities. When this section is complete, the program will be able to save the cost information of booked vacation packages to a reservation file. It will also display a list of all the data stored in the file. The menu that is displayed lists the four vacation packages, plus a fifth option which causes the data stored in the reservation file to be displayed. A sixth option exits the program.

New Modules

Three new modules, or functions, will be added to the program: `openFile`, `saveInfo`, and `showRes`. Table 1 describes the purpose of each module.

Table 1

Module Name	Description
`openFile`	Opens the requested file.
`saveInfo`	Saves the cost data of a vacation package reservation to the file currently open.
`showRes`	Displays all of the package reservation data in the file currently open.

The `openFile` Module

The `OpenFile` module is called prior to the menu being displayed. It first asks the user to enter the name of a file. That file is then opened for input and output, in binary mode. Here is the pseudocode:

```
openFile Module
   Ask user for file name.
   Open file in binary mode for input and output.
   If file failed to open
      Create the file.
   End If.
End Module.
```

The C++ code is shown here:

```
void openFile(fstream &file)
{
   const int SIZE = 256;
   char fileName[SIZE];

   cout << "File name: ";
   cin.getline(fileName, SIZE);

   file.open(fileName, ios::in | ios::out | ios::binary);
   if (!file)
   {
      cout << "Creating " << fileName << "...\n";
      // Create the file.
      file.open(fileName, ios::out);
      // Close the file.
      file.close();
      // Reopen the file for input and output.
      file.open(fileName, ios::in | ios::out | ios::binary);
   }
}
```

The `saveInfo` Module

The `saveInfo` module is called after the costs for a specific vacation package have been calculated. The user is asked if he or she wants to save the data. If so, the data structure holding the data is written to the reservation file currently open. Here is the pseudocode:

```
saveInfo Module
   Ask user "Do you want to save this data?".
   If "yes"
      Write record to the file.
      If write operation failed
         Display error message.
      End If.
   End If.
End Module.
```

The C++ code is shown here:

```
void saveInfo(Reservation &group, fstream &file)
{
   char yorN;

   cout << "Do you want to save this data? (Y/N) ";
   cin >> yorN;
   yorN = toupper(yorN);
```

```
    // Validate input
    while (yorN != 'Y' && yorN != 'N')
    {
       cout << "Please enter Y or N\n";
       cin >> yorN;
       yorN = toupper(yorN);
    }

    // Save the data.
    if (yorN == 'Y')
    {
       cout << "Saving reservation data.\n";
       file.write(reinterpret_cast<char *>(&group), sizeof(group));
       if (!file)
          cout << "Could not write to file!\n";
    }
}
```

The showRes Module

When the user selects option 5 from the menu, the showRes module is called. This function moves the file's read position to the beginning of the file. It then begins a loop that displays the record's data, and reads the next record from the file. The loop repeats until the end of the file is encountered. Here is the pseudocode:

```
showRes Module
       Seek beginning of file.
       Read a record from the file.
       While not at the end of the file
          Display the record.
          Ask user to press a key to continue.
          Read a record from the file.
       End While.
End Module.
```

The C++ code is shown here:

```
void showRes(fstream &file)
{
   Reservation temp;
   char skip[2];

   file.seekg(0L, ios::beg);  // Go to beginning of file.
   file.read(reinterpret_cast<char *>(&temp), sizeof(temp));
   while (!file.eof())
   {
      displayInfo(temp);
      cout << "Type a character and press Enter "
           << "to continue:";
      cin >> skip;
      file.read(reinterpret_cast<char *>(&temp), sizeof(temp));
   }
   if (file.fail())
      file.clear();           // Clear any error state
}
```

The function defines a local variable, `temp`, which is a `Reservation` structure. It's used to hold each record as it is read from the file. After each record's data is displayed, the message `"Type a character and press Enter"` is displayed. When the user performs this action, the function repeats the loop, reading the next record.

The last `if` statement tests the condition of the file's `fail` bit. If `file.fail` returns true, the `fail` bit is cleared so processing may resume. (The `fail` bit may be set after the last record has been read, as a result of the read member trying to read past the end of the file.)

Program 12-23 shows the entire program.

Program 12-23

```
1  // This program will assist the High Adventure Travel Agency
2  // in booking reservations for any of their 4 major
3  // vacation packages.
4  #include <iostream>
5  #include <fstream>
6  #include <iomanip>
7  #include <cctype>
8  using namespace std;
9
10 // Data Structures
11 struct Package1            // Climbing Package
12 {
13    int      num;           // Number in party
14    int      beginners;     // Those needing instruction
15    int      advanced;      // Those not needing instruction
16    int      needEquip;     // Those renting camping equipment
17    double   baseCharges;   // Base charges
18    double   charges;       // Total charges
19    double   instruction;   // Cost of instruction
20    double   equipment;     // Cost of equipment rental
21    double   discount;      // Discount
22    double   deposit;       // Required deposit
23 };
24
25 struct Package2            // Scuba Package
26 {
27    int      num;           // Number in party
28    int      beginners;     // Those needing instruction
29    int      advanced;      // Those not needing instruction
30    double   baseCharges;   // Base charges
31    double   charges;       // Total charges
32    double   instruction;   // Cost of instruction
33    double   discount;      // Discount
34    double   deposit;       // Required deposit
35 };
36
```

(program continues)

Program 12-23 *(continued)*

```
37 struct Package3            // Sky Diving Package
38 {
39    int      num;           // Number in party
40    int      lodge1;        // Number at 1st lodging choice
41    int      lodge2;        // Number at 2nd lodging choice
42    double   baseCharges;   // Base charges
43    double   charges;       // Total charges
44    double   discount;      // Discount
45    double   lodging;       // Cost of lodging
46    double   deposit;       // Required deposit
47 };
48
49 struct Package4            // Spelunking Package
50 {
51    int      num;           // Number in party
52    int      needEquip;     // Those renting camping equipment
53    double   baseCharges;   // Base charges
54    double   charges;       // Total charges
55    double   equipment;     // Cost of equipment rental
56    double   discount;      // Discount
57    double   deposit;       // Required deposit
58 };
59
60 union Pack
61 {
62    struct Package1 climb;
63    struct Package2 scuba;
64    struct Package3 sky;
65    struct Package4 spel;
66 };
67
68 struct Reservation
69 {
70    int packNum;
71    union Pack packs;
72 };
73
74 // Constants for the charges.
75 const double CLIMB_RATE = 350.0;       // Base rate - Devil's Courthouse
76 const double SCUBA_RATE = 1000.0;      // Base rate - Bahamas
77 const double SKY_DIVE_RATE = 400.0;    // Base rate - Sky diving
78 const double CAVE_RATE = 700.0;        // Base rate - Spelunking
79 const double CLIMB_INSTRUCT = 100.0;   // Climbing instruction
80 const double SCUBA_INSTRUCT = 100.0;   // Scuba instruction
81 const double DAILY_CAMP_RENTAL = 40.0; // Daily camping equip. rental
82 const double DAY_LODGE_1 = 65.0;       // Lodging option (sky diving)
83 const double DAY_LODGE_2 = 120.0;      // Lodging option (sky diving)
84
```

(program continues)

Program 12-23 *(continued)*

```
// Function prototypes
void openFile(fstream &);
void saveInfo(Reservation &, fstream &);
void climbing(Reservation &);
void scuba(Reservation &);
void skyDive(Reservation &);
void spelunk(Reservation &);
int  menu();
void displayInfo(Reservation &);
void displayPack1(Reservation &);
void displayPack2(Reservation &);
void displayPack3(Reservation &);
void displayPack4(Reservation &);
void showRes(fstream &);

int main()
{
   int selection;
   Reservation group;
   fstream file;

   cout << fixed << showpoint << setprecision(2);
   openFile(file);
   do
   {
     selection = menu();
     switch(selection)
     {
        case 1 :  climbing(group);
                  break;
        case 2 :  scuba(group);
                  break;
        case 3 :  skyDive(group);
                  break;
        case 4 :  spelunk(group);
                  break;
        case 5 :  showRes(file);
                  break;
        case 6 :  cout << "Exiting program.\n\n";
     }
     if (selection < 5)
    {
      displayInfo(group);
      saveInfo(group, file);
     }
   } while (selection != 6);
   file.close();
   return 0;
}
```

(program continues)

Program 12-23 *(continued)*

```
135 //************************************************
136 // Definition of function openFile.               *
137 // Accepts an fstream object as an argument. The  *
138 // file is opened for both input and output, in   *
139 // binary mode.                                   *
140 //************************************************
141
142 void openFile(fstream &file)
143 {
144    const int SIZE = 256;
145    char fileName[SIZE];
146
147    cout << "File name: ";
148    cin.getline(fileName, SIZE);
149
150    file.open(fileName, ios::in | ios::out | ios::binary);
151    if (!file)
152    {
153       cout << "Creating " << fileName << "...\n";
154       // Create the file.
155       file.open(fileName, ios::out);
156       // Close the file.
157       file.close();
158       // Reopen the file for input and output.
159       file.open(fileName, ios::in | ios::out | ios::binary);
160    }
161 }
162
163 //*********************************************************
164 // Definition of function saveInfo.                       *
165 // Accepts a Reservation structure and an fstream object.  *
166 // The user is asked if the data in the structure          *
167 // is to be saved. If so, it is saved at the end of the file.*
168 //*********************************************************
169
170 void saveInfo(Reservation &group, fstream &file)
171 {
172    char yorN;
173
174    cout << "Do you want to save this data? (Y/N) ";
175    cin >> yorN;
176    yorN = toupper(yorN);
177
```

(program continues)

Program 12-23 *(continued)*

```
178    // Validate input
179    while (yorN != 'Y' && yorN != 'N')
180    {
181       cout << "Please enter Y or N\n";
182       cin >> yorN;
183       yorN = toupper(yorN);
184    }
185
186    // Save the data.
187    if (yorN == 'Y')
188    {
189       cout << "Saving reservation data.\n";
190       file.write(reinterpret_cast<char *>(&group), sizeof(group));
191       if (!file)
192          cout << "Could not write to file!\n";
193    }
194 }
195
196 //*************************************************
197 // Definition of function menu.                   *
198 // Displays the main menu and asks the user to select *
199 // an option. Returns an integer in the range 1 - 6.  *
200 //*************************************************
201
202 int menu( )
203 {
204     int choice;
205
206    cout << "High Adventure Travel Agency\n";
207    cout << "----------------------------\n";
208    cout << "1) Devil's Courthouse Adventure Weekend\n";
209    cout << "2) Scuba Bahama\n";
210    cout << "3) Sky Dive Colorado\n";
211    cout << "4) Barron Cliff Spelunk\n";
212    cout << "5) Show Booked Reservations\n";
213    cout << "6) Exit Program\n\n";
214    cout << "Enter 1, 2, 3, 4, 5, or 6: ";
215    cin >> choice;
216
217     while (choice < 1 || choice > 6)
218     {
219       cout << "Invalid Selection\n";
220       cin >> choice;
221     }
222     return choice;
223 }
224
```

(program continues)

Program 12-23 *(continued)*

```
225 //*****************************************************
226 // Definition of climbing function.                   *
227 // Uses a Reservation reference parameter to hold the  *
228 // vacation package information.                       *
229 // This function calculates the charges for the        *
230 // Devil's Courthouse Adventure Weekend package.       *
231 //*****************************************************
232
233 void climbing(Reservation &group)
234 {
235    group.packNum = 1;
236    cout << "\nDevil's Courthouse Adventure Weekend\n";
237    cout << "-----------------------------------\n";
238    cout << "How many will be going who need an instructor? ";
239    cin >> group.packs.climb.beginners;
240    cout << "How many advanced climbers will be going? ";
241    cin >> group.packs.climb.advanced;
242    group.packs.climb.num = group.packs.climb.beginners +
243              group.packs.climb.advanced;
244    cout << "How many will rent camping equipment? ";
245    cin >> group.packs.climb.needEquip;
246    // Calculate base charges.
247    group.packs.climb.baseCharges = group.packs.climb.num *
248                 CLIMB_RATE;
249    group.packs.climb.charges = group.packs.climb.baseCharges;
250    // Calculate 10% discount for 5 or more.
251    if (group.packs.climb.num > 4)
252    {
253       group.packs.climb.discount = group.packs.climb.charges
254                 * .1;
255       group.packs.climb.charges -= group.packs.climb.discount;
256    }
257    else
258       group.packs.climb.discount = 0;
259    // Add cost of instruction.
260    group.packs.climb.instruction = group.packs.climb.beginners
261                 * CLIMB_INSTRUCT;
262    group.packs.climb.charges += group.packs.climb.instruction;
263    // Add cost of camping equipment rental
264    group.packs.climb.equipment = group.packs.climb.needEquip *
265              DAILY_CAMP_RENTAL * 4;
266    group.packs.climb.charges += group.packs.climb.equipment;
267    // Calculate required deposit.
268    group.packs.climb.deposit = group.packs.climb.charges / 2.0;
269 }
270
```

(program continues)

Program 12-23 *(continued)*

```
271 //*****************************************************
272 // Definition of scuba function.                      *
273 // Uses a Reservation reference parameter to hold the  *
274 // vacation package information.                      *
275 // This function calculates the charges for the       *
276 // Scuba Bahama package.                              *
277 //*****************************************************
278
279 void scuba(Reservation &group)
280 {
281    group.packNum = 2;
282    cout << "\nScuba Bahama\n";
283    cout << "-----------------------------------\n";
284    cout << "How many will be going who need an instructor? ";
285    cin >> group.packs.scuba.beginners;
286    cout << "How many advanced scuba divers will be going? ";
287    cin >> group.packs.scuba.advanced;
288    group.packs.scuba.num = group.packs.scuba.beginners +
289             group.packs.scuba.advanced;
290    // Calculate base charges.
291    group.packs.scuba.baseCharges = group.packs.scuba.num *
292             SCUBA_RATE;
293    group.packs.scuba.charges = group.packs.scuba.baseCharges;
294    // Calculate 10% discount for 5 or more.
295    if (group.packs.scuba.num > 4)
296    {
297      group.packs.scuba.discount = group.packs.scuba.charges
298            * .1;
299      group.packs.scuba.charges -= group.packs.scuba.discount;
300    }
301    else
302       group.packs.scuba.discount = 0;
303    // Add cost of instruction.
304    group.packs.scuba.instruction = group.packs.scuba.beginners
305           * SCUBA_INSTRUCT;
306    group.packs.scuba.charges += group.packs.scuba.instruction;
307    // Calculate required deposit.
308    group.packs.scuba.deposit = group.packs.scuba.charges / 2.0;
309 }
310
311 //*****************************************************
312 // Definition of skyDive function.                    *
313 // Uses a Reservation reference parameter to hold the  *
314 // vacation package information.                      *
315 // This function calculates the charges for the       *
316 // Sky Dive Colorado package.                         *
317 //*****************************************************
318
```

(program continues)

Program 12-23 *(continued)*

```
319 void skyDive(Reservation &group)
320 {
321    group.packNum = 3;
322    cout << "\nSky Dive Colorado\n";
323    cout << "-----------------------------------\n";
324    cout << "How many will be going? ";
325    cin >> group.packs.sky.num;
326    // Calculate base charges.
327    group.packs.sky.baseCharges = group.packs.sky.num *
328               SKY_DIVE_RATE;
329    group.packs.sky.charges = group.packs.sky.baseCharges;
330    // Calculate 10% discount for 5 or more.
331    if (group.packs.sky.num > 4)
332    {
333      group.packs.sky.discount = group.packs.sky.charges * .1;
334      group.packs.sky.charges -= group.packs.sky.discount;
335    }
336    else
337       group.packs.sky.discount = 0;
338    // Calculate lodging costs.
339    cout << "How may will stay at Wilderness Lodge? ";
340    cin >> group.packs.sky.lodge1;
341    cout << "How many will stay at Luxury Inn? ";
342    cin >> group.packs.sky.lodge2;
343    group.packs.sky.lodging = (group.packs.sky.lodge1 *
344        DAY_LODGE_1) + (group.packs.sky.lodge2 * DAY_LODGE_2);
345     group.packs.sky.charges += group.packs.sky.lodging;
346    // Calculate required deposit.
347    group.packs.sky.deposit = group.packs.sky.charges / 2.0;
348 }
349
350 //*****************************************************
351 // Definition of spelunk function.                    *
352 // Uses a Reservation reference parameter to hold the  *
353 // vacation package information.                      *
354 // This function calculates the charges for the       *
355 // Barron Cliff Spelunk package.                      *
356 //*****************************************************
357
358 void spelunk(Reservation &group)
359 {
360    group.packNum = 4;
361    cout << "\nBarron Cliff spelunk Weekend\n";
362    cout << "-----------------------------------\n";
363    cout << "How many will be going? ";
364    cin >> group.packs.spel.num;
365    cout << "How many will rent camping equipment? ";
366    cin >> group.packs.spel.needEquip;
```

(program continues)

Program 12-23 *(continued)*

```
367    // Calculate base charges.
368    group.packs.spel.baseCharges = group.packs.spel.num *
369              CAVE_RATE;
370    group.packs.spel.charges = group.packs.spel.baseCharges;
371    // Calculate 10% discount for 5 or more.
372    if (group.packs.spel.num > 4)
373    {
374      group.packs.spel.discount = group.packs.spel.charges * .1;
375      group.packs.spel.charges -= group.packs.spel.discount;
376    }
377    else
378       group.packs.spel.discount = 0;
379    // Add cost of camping equipment rental
380    group.packs.spel.equipment = group.packs.spel.needEquip *
381             DAILY_CAMP_RENTAL * 4;
382    group.packs.spel.charges += group.packs.spel.equipment;
383    // Calculate required deposit.
384    group.packs.spel.deposit = group.packs.spel.charges / 2.0;
385 }
386
387 //************************************************************
388 // Definition of function displayInfo.                       *
389 // Uses a Reservation reference parameter to hold the        *
390 // vacation package information. This function looks in the  *
391 // group.packNum member to determine which function to call  *
392 // to display the vacation package information.              *
393 //************************************************************
394
395 void displayInfo(Reservation &group)
396 {
397    switch (group.packNum)
398    {
399       case 1:  displayPack1(group);
400             break;
401       case 2:  displayPack2(group);
402             break;
403       case 3:  displayPack3(group);
404             break;
405       case 4:  displayPack4(group);
406             break;
407       default: cout << "ERROR: Invalid package number.\n";
408    }
409 }
410
411 //************************************************************
412 // Definition of function displayPack1.                      *
413 // Uses a Reservation reference parameter to hold the        *
414 // vacation package information. This function displays the  *
415 // information stored for vacation package 1.                *
416 //************************************************************
417
```

(program continues)

Program 12-23 *(continued)*

```
418 void displayPack1(Reservation &group)
419 {
420    cout << "Package: Devil's Courthouse Adventure Weekend\n";
421    cout << "Number in party:  "
422       << group.packs.climb.num << endl;
423    cout << "Base charges: $"
424       << group.packs.climb.baseCharges << endl;
425    cout << "Instruction cost: $"
426       << group.packs.climb.instruction << endl;
427    cout << "Equipment rental: $"
428       << group.packs.climb.equipment << endl;
429    cout << "Discount: $"
430       << group.packs.climb.discount << endl;
431    cout << "Total charges: $"
432       << group.packs.climb.charges << endl;
433    cout << "Required deposit: $"
434       << group.packs.climb.deposit << endl << endl;
435 }
436
437 //************************************************************
438 // Definition of function displayPack2.                      *
439 // Uses a Reservation reference parameter to hold the        *
440 // vacation package information. This function displays the  *
441 // information stored for vacation package 2.                *
442 //************************************************************
443
444 void displayPack2(Reservation &group)
445 {
446    cout << "Package: Scuba Bahama\n";
447    cout << "Number in party:  "
448       << group.packs.scuba.num << endl;
449    cout << "Base charges: $"
450       << group.packs.scuba.baseCharges << endl;
451    cout << "Instruction cost: $"
452       << group.packs.scuba.instruction << endl;
453    cout << "Discount: $"
454       << group.packs.scuba.discount << endl;
455    cout << "Total charges: $"
456       << group.packs.scuba.charges << endl;
457    cout << "Required deposit: $"
458       << group.packs.scuba.deposit << endl << endl;
459 }
460
461 //************************************************************
462 // Definition of function displayPack3.                      *
463 // Uses a Reservation reference parameter to hold the        *
464 // vacation package information. This function displays the  *
465 // information stored for vacation package 3.                *
466 //************************************************************
467
```

(program continues)

Program 12-23 *(continued)*

```
468 void displayPack3(Reservation &group)
469 {
470    cout << "Package: Sky Dive Colorado\n";
471    cout << "Number in party:  "
472       << group.packs.sky.num << endl;
473    cout << "Base charges: $"
474       << group.packs.sky.baseCharges << endl;
475    cout << "Lodging: $"
476       << group.packs.sky.lodging << endl;
477    cout << "Discount: $"
478       << group.packs.sky.discount << endl;
479    cout << "Total charges: $"
480       << group.packs.sky.charges << endl;
481    cout << "Required deposit: $"
482       << group.packs.sky.deposit << endl << endl;
483 }
484
485 //************************************************************
486 // Definition of function displayPack4.                      *
487 // Uses a Reservation reference parameter to hold the        *
488 // vacation package information. This function displays the  *
489 // information stored for vacation package 4.                *
490 //************************************************************
491
492 void displayPack4(Reservation &group)
493 {
494    cout << "Package: Barron Cliff Spelunk\n";
495    cout << "Number in party:  "
496       << group.packs.spel.num << endl;
497    cout << "Base charges: $"
498       << group.packs.spel.baseCharges << endl;
499    cout << "Equipment rental: $"
500       << group.packs.spel.equipment << endl;
501    cout << "Discount: $"
502       << group.packs.spel.discount << endl;
503    cout << "Total charges: $"
504       << group.packs.spel.charges << endl;
505    cout << "Required deposit: $"
506       << group.packs.spel.deposit << endl << endl;
507 }
508
509 //************************************************************
510 // Definition of function showRes.                           *
511 // Accepts an fstream object as an argument. Seeks the       *
512 // beginning of the file and then reads and displays         *
513 // each record.                                              *
514 //************************************************************
515
```

(program continues)

Program 12-23 *(continued)*

```
516 void showRes(fstream &file)
517 {
518    Reservation temp;
519    char skip[2];
520
521    file.seekg(0L, ios::beg);  // Go to beginning of file.
522    file.read(reinterpret_cast<char *>(&temp), sizeof(temp));
523    while (!file.eof())
524    {
525       displayInfo(temp);
526       cout << "Type a character and press Enter "
527          << "to continue:";
528       cin >> skip;
529       file.read(reinterpret_cast<char *>(&temp), sizeof(temp));
530    }
531    if (file.fail())
532       file.clear();  // Clear any error state
533 }
```

Program Output with Example Input Shown in Bold

```
File name: resfile [Enter]
High Adventure Travel Agency
----------------------------
1) Devil's Courthouse Adventure Weekend
2) Scuba Bahama
3) Sky Dive Colorado
4) Barron Cliff Spelunk
5) Show Booked Reservations
6) Exit Program

Enter 1, 2, 3, 4, 5, or 6: 1 [Enter]

Devil's Courthouse Adventure Weekend
------------------------------------
How many will be going who need an instructor? 3 [Enter]
How many advanced climbers will be going? 2 [Enter]
How many will rent camping equipment? 3 [Enter]
Package: Devil's Courthouse Adventure Weekend
Number in party:  5
Base charges: $1750.00
Instruction cost: $300.00
Equipment rental: $480.00
Discount: $175.00
Total charges: $2355.00
Required deposit: $1177.50

Do you want to save this data? (Y/N) y [Enter]
Saving reservation data.
```

(program output continues)

Program 12-23 *(continued)*

```
High Adventure Travel Agency
----------------------------
1) Devil's Courthouse Adventure Weekend
2) Scuba Bahama
3) Sky Dive Colorado
4) Barron Cliff Spelunk
5) Show Booked Reservations
6) Exit Program

Enter 1, 2, 3, 4, 5, or 6: 3 [Enter]

Sky Dive Colorado
-----------------------------------
How many will be going? 8 [Enter]
How many will stay at Wilderness Lodge? 4 [Enter]
How many will stay at Luxury Inn? 4 [Enter]
Package: Sky Dive Colorado
Number in party:  8
Base charges: $3200.00
Lodging: $740.00
Discount: $320.00
Total charges: $3620.00
Required deposit: $1810.00

Do you want to save this data? (Y/N) y [Enter]
Saving reservation data.

High Adventure Travel Agency
----------------------------
1) Devil's Courthouse Adventure Weekend
2) Scuba Bahama
3) Sky Dive Colorado
4) Barron Cliff Spelunk
5) Show Booked Reservations
6) Exit Program

Enter 1, 2, 3, 4, 5, or 6: 5 [Enter]

Package: Devil's Courthouse Adventure Weekend
Number in party:  5
Base charges: $1750.00
Instruction cost: $300.00
Equipment rental: $480.00
Discount: $175.00
Total charges: $2355.00
Required deposit: $1177.50
```

(program output continues)

Program 12-23 *(continued)*

```
Type a character and press Enter to continue: g [Enter]
Package: Sky Dive Colorado
Number in party:  8
Base charges: $3200.00
Lodging: $740.00
Discount: $320.00
Total charges: $3620.00
Required deposit: $1810.00

Type a character and press Enter to continue: g [Enter]

High Adventure Travel Agency
----------------------------
1) Devil's Courthouse Adventure Weekend
2) Scuba Bahama
3) Sky Dive Colorado
4) Barron Cliff Spelunk
5) Show Booked Reservations
6) Exit Program

Enter 1, 2, 3, 4, 5, or 6: 6 [Enter]
Exiting program.
```