**IS 350—Prof. Mark G. Simkin Case 3: Downhill Snowboard Shop**

**Draft 1**



**Objective:** The purpose of this assignment is to create a Visual Basic project that helps a snowboard shop prepare customer invoices. Although the final invoice screen is straightforward, there are several supporting screens, and the entire project will take time to complete. Therefore, you are encouraged to work in teams of size 3 and to start early! *Three important requirements of this case are: (1) to use control arrays for all purchase screens, (2) to allow blanks for non-used fields which must remain blank, and (3) to use your own background colors and icons.*

**Password Screen.** The first screen the user sees is the Password Screen in Figure A. User passwords are case sensitive. These are read from a sequential file in sets of four items each, separated by commas (i.e., from a CSV file) that will be provided to you for this case. The format is Last Name, User ID, Password, and Creation Date—for example:

Able, AND, Apples, 4/2/11

Baker, BAKER, Babboon, 3/3/10

Carson, CARSON, Catman99, 4/4/09

Etc.

Note that user last names start with a capital letter, while the User IDs are all in capital letters. As illustrated in Figure A, the system allows users to enter their User IDs in lower case. At startup, the system stores these items in separate data arrays and verifies a user by matching his or her User ID and password with values in the arrays. If the user fails to provide a correct user ID and associated password in three tries, or after 30 seconds, the system alerts him or her to the problem and terminates execution. (Hint: use a form load event to start a timer that clocks the user.) Finally, if the user clicks on the “Clear Entries” button, the system erases the contents of the two textboxes and allows the user to try again. (The system does not reset the timer if the user clicks this button.)

**Main Screen.** If a user logs in successfully, the system displays the Main screen in Figure 1. Note that this screen displays (1) the system date (using DateTime.Today and a Form\_Load event) as the default date (which the user can change at run time) and (2) a series of textboxes for the customer’s name, address, etc., as well as an (optional) Email address. The Main screen also allows the user to access several other screens (described below). Regardless of which alternate screen the user might like to see (except for the About, Quit, and Administration buttons), the system uses a custom function to validate the customer information in the form. If any textbox is blank, including the date but excluding the optional email address, the system displays a message box informing the user of the specific missing data item, sets the focus to the blank text box, and does not display an alternate form. The system also uses separate global-level ***subroutines***(that return Boolean values) to check for a valid phone number or zip code.

**Shipping Information Screen.** If the customer’s shipping address differs from the address entered in the Main screen, the user can click on the “New Shipping Address” button in the main form. This displays the Shipping Information screen in Figure 2, which allows the user to enter different address information. The system uses the “Main Screen” button click event to validate the information in the shipping screen, which must include calls to the custom validation functions described above that check for a valid phone number or zip code. If all data are valid when the user clicks on the Main Screen button, the system closes the Shipping Information screen. The user returns to the Main screen button by default.

**Snowboards Screen.** If the user clicks on the “Boards” button in the Main screen, the system displays the Snowboards screen shown in Figure 3. This screen allows the user to purchase snowboards by entering the quantity desired in the text boxes provided. The prices for the various boards are obtained from a sequential file that initializes the appropriate labels in the screen at the time the screen loads. Note that there is also a special “Running total” label in the lower-right corner of the screen. If the Snowboards screen is the first screen accessed, the running total value will be zero. After the user makes some purchasing choices and clicks on the “Calculate Total” button, the system computes the total dollar value of snowboard purchases and updates the running total, as shown in the figure. If the user clicks on the Clear button, the system erases the contents of the text boxes and returns the focus to the first text box. If the user clicks on the “Main Screen” button, the system closes the Snowboards screen and displays the Main screen. If the user clicks on the Exit button, the system ends execution.

The managers of the Downhill Snowboard Shop are considering only carrying products with preprinted UPC codes. They have decided to experiment by first trying UPC codes only on snowboards. To make their system work, the managers have already commissioned someone to add the new, top row of entries in this screen. This enables salesclerks to enter a new type of snowboard (with a new price) into the system if a customer buys one (or more) of them.

The application must validate the UPC codes entered. To illustrate, suppose the UPC code is 0-64200-11589-6. The validation steps are as follows:

1. Strip the UPC code of unwanted dashes or any other special character. Here, the stripped number is 064200115896.
2. Validate that the length of the stripped code is 12. In this example, the length is 12 and they are all numeric, as required.
3. Treat the last digit as a check digit. Here, the last digit is “6” so this is the check digit.
4. Add the odd-position digits (=0+4+0+1+5+9 = 19) and multiply this sum by 3. (Here: 19 × 3 = 57.)
5. Add the even-position digits (=6+2+0+1+8 = 17).
6. Add the values in steps 4 and 5 together. (Here, the sum is 57+17 = 74).
7. Examine the last digit of the sum. If it is 0, the computed check digit is also 0. If the last digit is not zero, subtract this digit from 10 and use this number as the computed check digit. Here, the last digit in the sum is “4,” and the check digit is therefore 10 - 4 = 6.
8. Compare the computed check digit found in step 7 to check digit found in step 3. If they match, we conclude the UPC code is valid. If they do not match, we conclude the opposite. Here, the computed value of “6” matches the check digit of “6” in step 3, and we therefore conclude that this UPC code is valid.

The system need only validate the UPC entry for a UPC entry in the top row. You can hard code the UPC values into the other textboxes as shown, and it is not necessary to validate them. If a UPC code is not valid, the system should output an error message, set the focus to the UPC textbox, and *not* compute a grand total.

**Snowboard Bindings Screen.** If the user clicks on the “Bindings” button in the Main screen, the system displays the Snowboard Bindings screen in Figure 4. The prices for the various bindings are obtained from a sequential file that initializes the appropriate labels in the screen at the time the screen loads. The textboxes, labels, and buttons in this screen work similar to those in the Snowboards screen. At present, the shop is having an “almost-two-for-one” sale on bindings. For each type of binding, the first set is full price and the second set is $1 more. The sale only applies to the first two sets. Thus, for example, if the price of one set of bindings is $179 and the user buys 3 sets, the total price is $359. See Figure 4 for other examples. If the user qualifies for a discount, the system displays a “Sales Price!” label to the right of the extension for that item. (Hint: use a second label array for this task.)

**Snowboard Boots Screen.** If the user clicks on the “Boots” button in the Main screen, the system displays the Snowboard Boots screen in Figure 5. The prices for the various boots are obtained from a sequential file that initializes the various labels in the screen at the time the screen loads. Note that this screen differs slightly from the Snowboards or Snowboard Bindings screens in that the user must input a “size” as well as a “quantity” for each pair of boots desired. Note also the label at the bottom of the screen, which explains that all boots are available in sizes 6-12, except Viking (restricted to sizes 7-11), Nordika (sizes 8-12), and Kids Fischer (sizes 3-9). If the user accidentally enters a boot size that falls outside these ranges, the system displays an error message in a message box, sets the focus to the first error it encounters, and does not calculate a final total. The code in the click procedure of the “Calculate Total” button should call a separate, custom subroutine (not function) to test for proper boot sizes. Finally, the system should also compute a running total of purchases for the current session (this includes boots, bindings, snowboards, and season passes, but not ski lessons).

**Season Passes Screen.** The snowboard shop offers discount season passes to the Visual Valley Ski Resort. If the user clicks on the “Season Pass” button on the Main screen, the system displays the Season Passes screen in Figure 6. This screen allows the user to order up to 4 tickets of each type shown. The system calculates prices using the date from the Main screen, which it displays in labels as shown in Figure 6. The code uses a Select Case statement to determine which column of values to use. To ensure usability from year to year, your code should only deal with month and day values, and ignore the year of purchase. (Hints: The Month(*Date*) function returns the number of the month from a date, and Microsoft.VisualBasic.DateAndTime.Day(*Date*) function returns the number of the day from a date.) If the user orders more than 4 tickets of any type, the system displays an error message and returns the user to the appropriate text box. The text boxes, labels, and buttons in this screen work similar to those in the other screens. (Note: you can store prices directly in the price labels in this form—no sequential file is required here.) Finally, note again that the system should also compute a running total of purchases for the current session (this includes boots, bindings, snowboards, and season passes, but not ski lessons).

**Lodging Screen.** In addition to purchasing season passes, the user can click on the “Lodging” button in the Season Passes screen to reserve a room at the Black Diamond Resort (see Figure 7). The user’s name (e.g., “Simon Snowboarder”) appears in red in a label at the time the form loads. The user selects the type of room, checkout preference, and special services desired. The charges for these items are as shown in the form (and which you can hard code in the application for this screen). The system validates that the user enters the desired check-in and check-out dates. If the user clicks on the Total Room Charges button, the system computes (1) the total days of the desired stay as the difference between the check-in and check-out dates, (2) the room costs (=room cost per day x number of days), and (3) all other costs (=the sum of any additional charges for checkout preferences and special services).

Check-out costs are charged only once, but all special services (in the upper-right portion of the screen) are charged for each day of the guest’s stay (and must therefore be multiplied by the number of days in the person’s stay). Currently, the resort is featuring a special discount: if the user selects five or more services (as illustrated here), the lowest-priced service (e.g., extra towels) is free. Programming hint: because there are only seven special services in total, the free item can only be one of three services. The system computes the correct charges and, if appropriate, indicates the free service for which the guest has qualified in a label for messages (middle lower portion of the form).

Finally, there is a surcharge for “high season” (i.e., the time period November 1 through March 31 and June 1 through August 30). (We will ignore overlapping stays between seasons here.) Regardless of whether the guest checks the “Check if in season” checkbox, the system tests for this (i.e., tests for months 1, 2, 3, 6, 7, 8, 11, and 12 using the “Date In” date). If the guest wishes a room during high season, *the system* should check the high season checkbox and add 10% to the guest’s bill as shown in Figure 7.

If the user clicks on the Clear button, the system erases all selections. If the user clicks on the Return to Season Pass screen, the system closes this form and transfers the total room charges to the bottom of the Season Pass screen. Finally, if the user clicks on the Exit button, the system ends execution.

**Lessons Screen.** Customers can also purchase discounted lessons from the Downhill Snowboard Shop. If the user clicks on the “Lessons” button in the Main screen, the system displays the Lessons screen in Figure 8. The prices for the various types of lessons are obtained from a sequential file that initializes the various labels in the screen at the time the screen loads. The Calculate Total, Clear, Main Screen, and Exit buttons work similar to those in the other screens with one exception.

For this screen only: if the user enters an incorrect quantity (e.g., a negative value), the system should turn the corresponding label (e.g., “Youth” to red). Finally, if the user purchases at least $100 in lessons and purchases lessons by November 30 (using the date from the main form), he or she will win a free lesson as shown in the yellow “Congratulations” label in Figure 8.

**Order Summary Screen.** When done ordering boards, boots, bindings, season passes, and lessons, the user can click on the “Check Out” button in the Main screen. The system then displays the Order Summary screen in Figure 8, which formats outputs to currency values. This screen uses labels to display a summary of the customer’s order. The customer’s first and last names that appear at the top of this form come from the Main screen, and code in the Form\_Load event should also compute the grand total as shown. If the user wishes to make changes to this order, he or she can click on the appropriate button, which displays the Snowboards, Snowboard Bindings, Snowboard Boots, Season Pass, or Lessons screens described above.

**Customer Invoice Screen.** If everything is satisfactory in the Order Summary screen, the user can click on the “Check Out” button in that screen. The system then displays the Customer Invoice screen in Figure 9. This screen uses labels to display all dollar information. The customer’s name and address comes from the Main screen. The “Amount of Order” field comes from the Order Summary screen.

The system should automatically compute the sales tax, which depends upon the state in the address field of the main form. The customer invoice should display the possible tax rates as shown in Figure 10, but can compute the tax from tax rates that you hard code into the click event for the Total Charges button. Note that, for Nevada, the tax only applies to purchases of snowboards, boots, bindings, ski passes, and lodging, but not lessons.

In the invoice of Figure 10, the cost for UPS shipping is $23.95 and for U.S. Mail shipping is $19.95. The cost for Federal Express Shipping is described below. The cost for insurance is $10 (Standard), $20 (Deluxe), or 3% (Full, computed as a percent of the amount of the order). Finally, the cost for Express Delivery is $29 additional and the cost for Saturday Delivery is $19 additional (both are possible). If the user clicks on *any* of the shipping objects (radio buttons or check boxes), the system changes the appropriate values as well as erases the “Total Charges” label.

When the user clicks on the “Total Charges” button, the system computes the total charges for the order. If the Shipping Charges or the Insurance Charges labels are blank, the system displays an error message in a message box, exits the procedure, and does *not* compute the total charges. This forces the user to click on the “Total Charges” button to re-compute the final charges. If the user clicks on the “Quit” button, execution ends. If the user clicks on the “Main Screen” button, the system closes the Customer Invoice screen and displays the Main form.

Finally, if the user clicks on the “Print Invoice” button, the system prints an invoice similar to the one in Figure 16. Note the underlining under some of the numbers, *which do not require a separate line*. Also note that the printed invoice differs slightly from the invoice in the form of Figure 10. Most of the information on this invoice comes directly from the application. Please make up your own invoice numbers, but be sure to use a different invoice number for each invoice you print.

**Federal Express Screen.** If the user clicks on the Federal Express radio button on the Customer Invoice screen, the system displays the Federal Express screen in Figure 11. This allows the customer to select a particular type of Federal Express service. Prices are as shown. Note that there is an extra charge for “morning delivery.” When the user selects a particular service and clicks on the “OK” button, the system displays the appropriate amount in the Shipping Charges label of the Invoice screen of Figure 10.

**Payment Screen.** When the user clicks on the “Total Charges” button in the Customer Invoice Screen and *after* performing any other validations pertinent to that screen, the system displays the Payment Screen shown in Figure 11. The combo box for the type of credit card type starts out blank. This forces the user to select one of four types of credit cards (Visa, Mastercard, American Express, or Discover) for payment from the combo box in the form.

If the user clicks on the Cancel button, the system redisplays the Customer Invoice Screen of Figure 10, but blanks out the label for “Total Charges,” thus forcing the user to click on the Total Charges button again in the Customer Invoice Screen. Note that the system allows the user to enter the credit card number with embedded dashes, blanks, or other special characters. Thus, the system must strip these extra characters from this entry to validate the credit card number.

In the Payments screen, if the user clicks on the OK button, the system first validates the information provided as outlined in the system requirements table below. Of special interest is how the system computes a “check digit” for the credit card that must match the expiration date’s month code. The computations are as follows: (1) Reading from left to right, multiply the digits in odd positions (1, 3, 5, etc.) by the fixed values 1, 2, 3, and so forth, and add them together. Call this Sum 1. (2) Add the digits in the even positions (2, 4, 6, etc.) Call this Sum 2. (3) Add Sum 1 + Sum 2. Call this Sum 3. (4) Add the digits in Sum 3 repeatedly to obtain a single check digit. (5) Take the month of the credit card’s expiration date. If it is a single digit, it is also the date check digit. If not, add the digits together to obtain a new date check digit. (6) Lastly, compare the two check digits. If they match, assume the credit card number is valid. If not, assume it is not valid. (Hint: Remember that Visual Basic starts counting from 0—not 1.) Note: For full credit, your application must use For-Next and/or Do loops to accomplish these tasks.

Here’s an example for credit card number 1234-5678-9012-3456 with expiration month code “11.”

**Position: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16**

**Multiples: 1 2 3 4 5 6 7 8**

**Card #: 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6**

Step 1: Sum 1 = 1\*1 + 3\*2 + 5\*3 + 7\*4 + 9\*5 + 1\*6 + 3\*7 + 5\*8 = 162

Step 2: Sum 2 = 2 + 4 + 6 + 8 + 0 + 2 + 4 + 6 = 32

Step 3: Sum 3 = 162 + 32 = 194

Step 4: Add the digits in Sum 3 to get 1+ 9 + 4 = 14. Add again to get 1 + 4 = 5.

Step 5: The expiration month is 11. The check digit is therefore 1 + 1 = 2.

Step 6: The check digit from step 4 does not match the check digit from step 5. Conclusion: the credit card number is not valid.

If the user fails to select a credit card choice from the combo box or enters data that fail any of the validation tests for this screen, the system should display a message box similar to the ones in Figures 13 and 14 and exit the sub procedure for the OK button. Note that the system displays the type of credit card in Figure 14, and for this reason, should be printed for each set of test data. If the data pass all the validation tests, the system should simply redisplay the Customer Invoice screen.

**Bonus: Password Maintenance Screen.** Complete this part of the project if you have time and wish to earn extra credit. The system administrator is able to add, delete, or modify password accounts using the Password Maintenance screen in Figure 15. Again, you should also use a textbox control array for the textboxes on the left side of this form. The administrator accesses this screen by clicking on the “Administration” button in the main form, and entering his or her personal administrative password (that you make up). Yes, this is a new password not associated with the processing involved with Figure A. Note: This password can be hard coded into the application. At form load for this screen, the system gives this person three tries to enter this password, and ends execution of the entire application if he or she fails the third attempt.

At form load, the system also reads all the password records from the password file into the (sorted) list box in the right side of the form. If the administrator clicks on the “Clear” button, the system erases the contents of the textboxes. If the administrator enters complete information for an end user and then clicks on the “Add to List” button, the system (1) makes sure that there is information in all four text boxes, (2) makes sure the first letter of the person’s last name is capitals while the rest of the name are in lower-case letters, (3 converts the User ID text to capitals, (4) adds the password information, complete with commas, to the list box, and (5) clears the textboxes.

If the administrator selects a record from the list box (by clicking on it) and then clicks on the “Delete” button, the system deletes that record from the list box. Note: The system should first make sure the administrator has in fact selected an item. If he or she has not done so, the system should provide a message box prompting for this, but not perform any further data processing.

Finally, if the administrator clicks on the “Return” button, the system first rewrites the password file by copying the contents of the list box to a temporary file, deleting the old password file, and renaming the temporary file with the name of the password file. The system then closes the Password Maintenance screen and the main form appears by default.

**System Requirements:** For this project, please note the following:

1. All buttons should have access keys.
2. The system allows back dating. Thus, the application should allow users to enter any valid date for 2015, and sell tickets at prices indicated in Figure 6.
3. The prices for snowboards, bindings, boots, and ski passes shown in the labels of Figures 3, 4, and 5 are obtained from separate (sequential) files. These values should be read using Form \_Load events and displayed in the price labels of the various screens. You should provide listings of these files, as well as a listing of the password file, in your documentation.
4. If there is no entry for a textbox, it should remain blank and not show “0.”
5. The application should use control arrays as described above.
6. All dollar figures should be right-justified. Also, only the values in the “Total” lines of the Snowboards, Snowboard Bindings, and Snowboard Boots screens and the dollar values in the Order Summary and Checkout screens should be formatted to currency values. All others should show dollars and cents, but no dollar signs.
7. The back colors for all screens should be consistent, and not the same as the one shown here. Feel free to pick something novel that appeals to your group.
8. Your code should validate the items listed in the table below.

|  |  |  |
| --- | --- | --- |
| **Form** | **Item** | **Requirement** |
| Main | Date | If the date is changed, it should be in the form m/d/y, where m is an integer between 1 and 12, d is an integer between 1 and 31, and y is any year greater than or equal to the current one. |
|  | Address entries | Last Name, Street Address, City, State, Zip Code, Country,  Phone Number are all required fields. |
|  | Zip code\* | Should be exactly five numeric digits. |
|  | Phone\* | Should be in the form (ddd) ddd-dddd, where the d’s are numeric values. There is one space embedded in the phone number, and validation should also test for the parentheses and the dash. |
| Shipping | Zip code\* | Same as zip code above |
|  | Phone\* | Same as phone above |
|  | Address entries | Same as for address entries above |
| Boards | Quantity fields | Must be blank or positive numbers. |
|  | UPC fields | Must satisfy the validation requirements described in the case. |
| Bindings | Quantity fields | Must be blank or positive numbers |
| Boots | Size Fields | Limited to sizes described in the case. |
|  | Quantity fields | Must be blank or positive numbers. |
| Season Pass | Quantity fields | Maximum of four passes for each type of ticket. |
|  | Total fields | Prices depend on the Date in the Main Screen |
| Lodging | Check-in/Check-out dates | Neither the check-in date nor the checkout date can be blank and the check-out date must be greater than the check-in date (tested in the code for the Total Room Charges button click procedure). |
|  | Checkout radio buttons | A checkout preference must be selected. |
| Lessons | Quantity fields | Must be blank or positive numbers. |
| Payment  Form | Text Box fields | Must not be blank |
|  | Card Number field | Each digit in the card number must be numeric, and its check digit must match the month code for the expiration date of the card, as explained in the text |
|  | Expiration date field | Must contain a valid month and year. The year must be greater than or equal to the year of the system date. If the years are equal, the expiration month of the card must be greater than system’s month. |
|  | Credit card type | A type of credit card must be selected from the combo box |
| Password Maintenance | Password values | Must not be blank |

**\***Validated in global subroutines. The phone number should contain parentheses as shown.

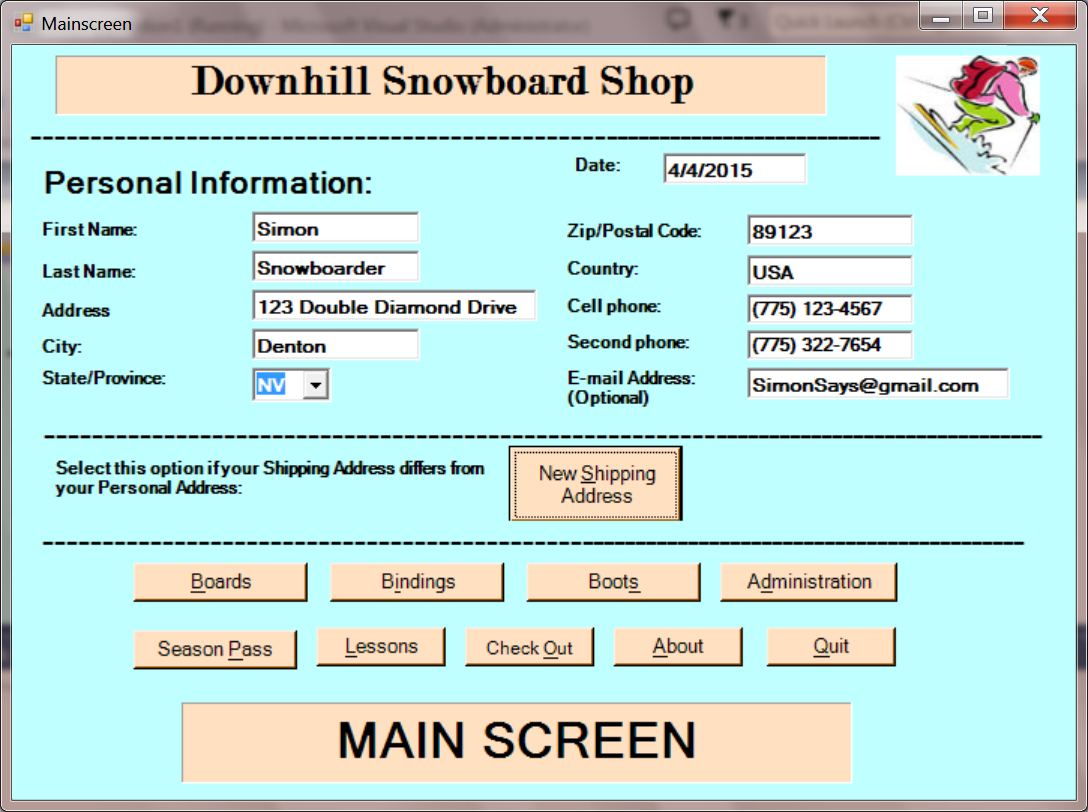
***dProject Requirements:***

This project requires a lot of work. To ensure the successful completion of this project, you should provide working copies of your forms to your team members for their review and approval one week prior to the due date of this project.

1. **Test Data**. Test your program with the data provided by your instructor. For each set of data, you should print a complete set of screen captures that show the resultant cost values. You should also print an invoice similar to the one in Figure 16 for each separate customer, and a separate payments screen similar to Figure 13 if the check digits for the credit card number and month code do not match. Collect these materials together, organized by test case, and label this set “Test Data.”
2. **Technical Documentation**. Create both internal and external technical documentation that describe your program in sufficient detail to allow someone else to understand it and maintain your code. To do this, provide (1) exemplary copies of all form images at run time, including an About form (not shown), (2) code for all forms in this assignment, and (3) a copy of each and every sequential file. The password file should contain at least ten sets of data in it (that you make up). All code should be clearly identified so that the reader knows what code corresponds to what form. Collect this set of pages together and label it “Technical Documentation.”
3. **User Manual.** The programmers who create a project are rarely the end users. Thus, you must also create a manual that helps non-technical individuals use your application. This (small) manual should include a general introduction and a description of what each button in each form does (see above). Collect this external documentation for your project together and label it “User Manual.”
4. **Enhancement.** This is your chance to be creative. Develop a single enhancement for this project that uses an additional form. *Documentation for this enhancement is critical.* Provide a screen capture of your form *at run time*, your VB code, and *carefully describe it on one or more separate pages*. Collect these pages together and label them “Enhancement.”
5. **Publication.** Combine the complete set of items described above in a small, 3-ring binder, complete with dividers identifying each part of your project. Other publication requirements are as follows:
   * The first, title page should contain the title of this project, the full names and pictures of the developers in your group, and a list of who did what tasks.
   * Each subsequent page should be numbered, and the first page following your title page should be a table of contents that indicates the starting page of each major section. Note: The code for each screen in your technical documentation is a section.
   * Your work should be printed on single-sided copy, *in color*, using a 12-point, Times Roman font.
   * Finally, use tabs to separate each the major part of your report, (i.e., starting with Part A and ending with Part D).
6. **Cheating Reminder:** Members from the same team can and should help each other. Students obtaining and using code, documentation, or other materials from others not on the same team is a violation of the UNR student code of conduct and will be treated as such.



**Figure A Password Screen**



**Figure 1 Main Screen**



**Figure 2 Shipping Screen**



**Figure 3 Snowboards Screen**



**Figure 4 Snowboard Bindings Screen**



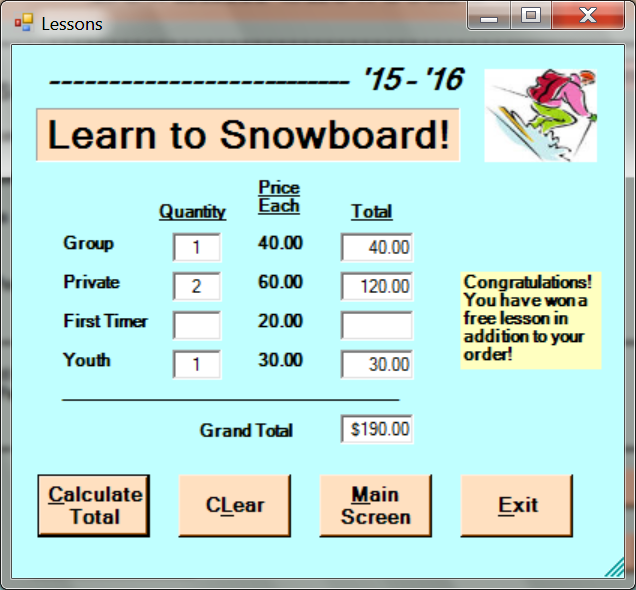
**Figure 5 Snowboard Boots Screen**



**Figure 6. Season Passes Screen**

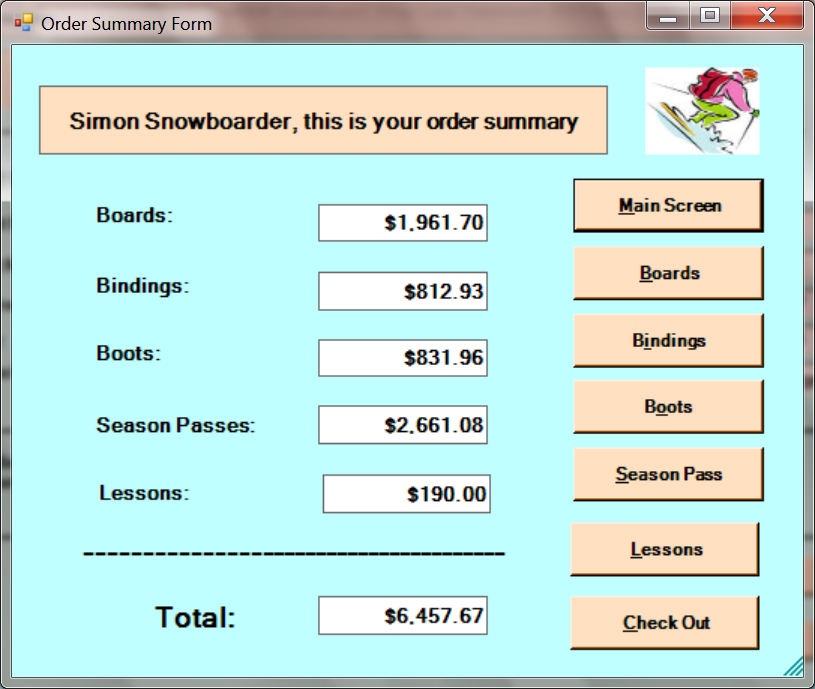


**Figure 7. Room Reservation Form**

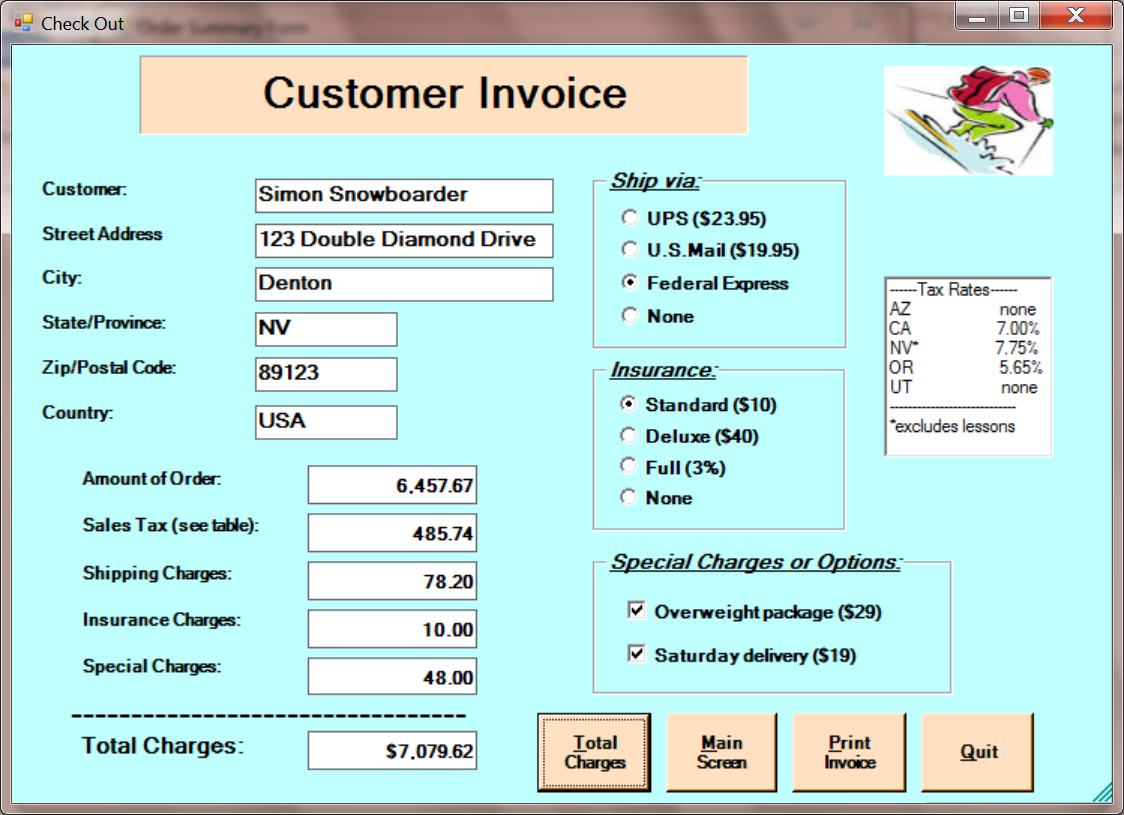


Bonus if user spends $100 or more and meets deadline.

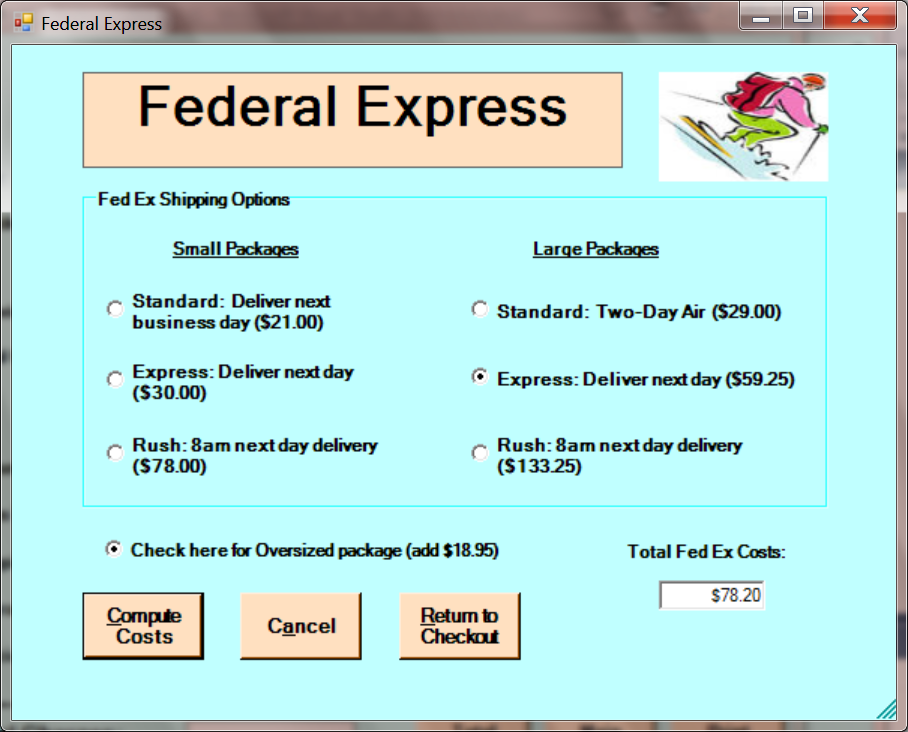
**Figure 8. Lessons Screen**



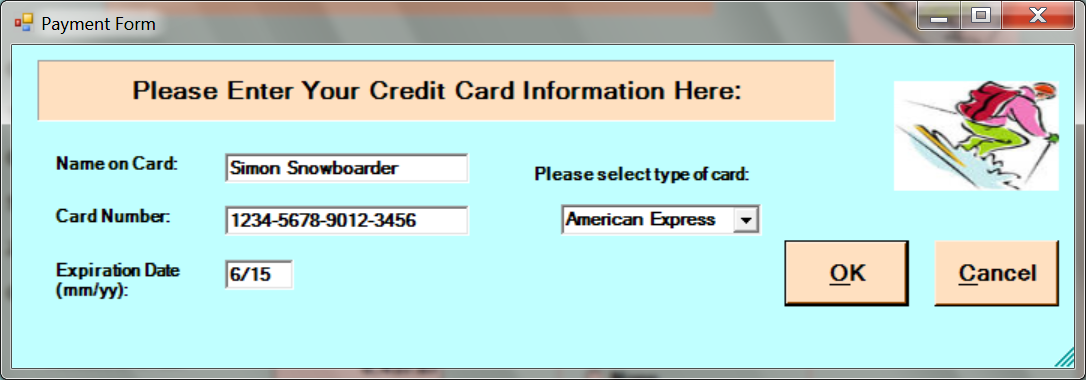
**Figure 9. Order Summary Screen**



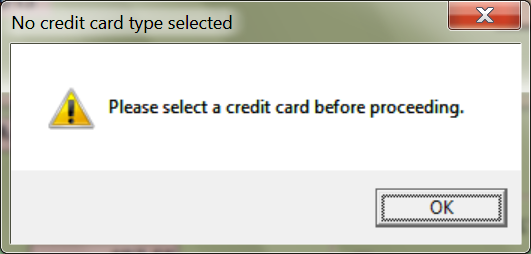
**Figure 10. Customer Invoice Screen (Federal Express charges are from Figure 11.) Note that the state of Nevada excludes labor such as ski lessons in the computation of sales tax.**



**Figure 11. Federal Express Screen**

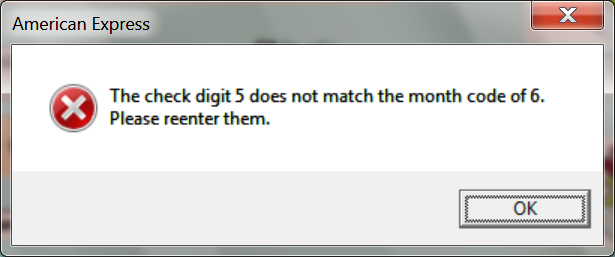


**Figure 12. Payment screen.**

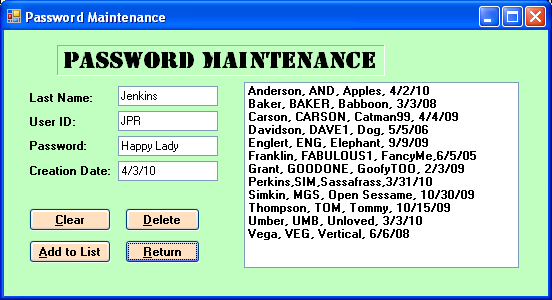


**Figure 13. An error message indicating that the user did not select a credit card**

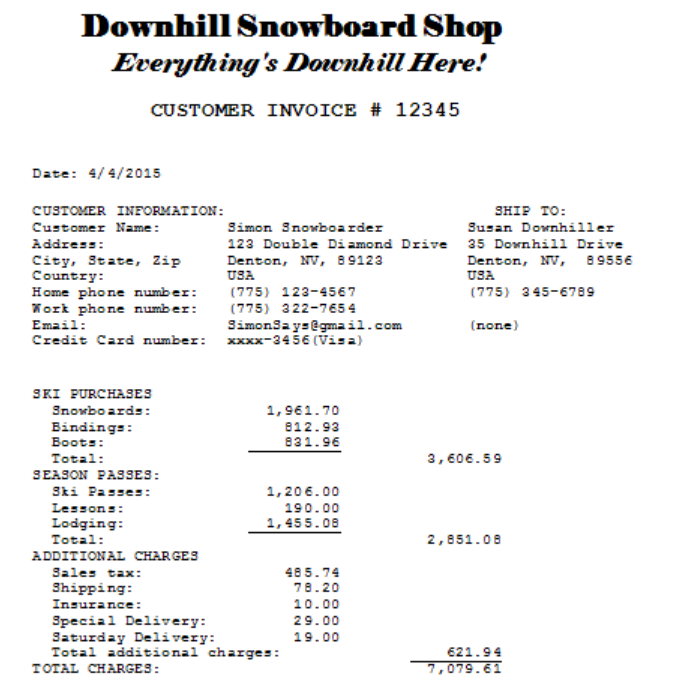
**from the combo box in the Payment screen**



**Figure 14. An error message indicating that the check digit computed for the credit card number did not match the month code. Be sure to include these numbers in your error message.**



**Figure 15. The password maintenance screen.**

****

**Figure 16. An example of a customer invoice. Please include underlining in your invoice.**