

SCHOOL OF COMPUTER SCIENCES

ACADEMIC SESSION 2016 / 2017, SEMESTER 2

CPT 113 – PROGRAMMING METHODOLOGY & DATA STRUCTURES

ASSIGNMENT 1

DUE DATE : 10th March 2017

NAME : CHONG KAR HIE

MATRIC NUMBER : 132893

GROUP NUMBER : 2 (D)

LECTURER’S NAME : MOHD AZAM OSMAN

**Content**

|  |  |
| --- | --- |
| Content | Page number |
| Introduction | 1 |
| Specification Requirements   * Inputs * Outputs * Process/formula * Constraints | 2  2  3-5  5 |
| Design of the program   * UML diagram and inheritance/composition hierarchy * Class description * Selected function description | 6  7-8  8 |
| C++ code (source code) | 9-21 |
| * Test Data * Print screen of sample input/output | 22  23-32 |

**Introduction**

The main purpose of this assignment is to help a company called “Dream Vacation Company” to design a program called “MyDreamV” which is vacation cost estimator based on the preference of customers when they are using the program. Nations in our country are busying working and yet to have time enjoying their life. Having a vacation has a lot of advantages like improve health, reduce stress, and the most important is, improve productivity. Dream Vacation Company offers many vacation packages for families or groups of people like friends in order to let their customers choose when and where they want to go for their preferred vacation. So, this program is created to help people in country to choose whichever place and duration they want to have a short run away from their work or other things that might made them stressed all the time. The company provides variety of vacation packages and customer can choose what and where they want freely. Customers can ether choose their desired region, season, package code, type of package, type of vacation, duration of vacation, and number of adults and children who will go for the vacation. The program will then help customers to calculate the total basic price and also the total price after adding GST in order to let customers to know the price so that they can have a budget for their vacation first. Thus, they can enjoy their vacation based on their chosen packages.

**Specification of Requirements:**

**Inputs:-**

User’s input:-

1. Number of adult
2. Number of children
3. Region of vacation
4. Duration of vacation
5. Vacation type
6. Package type
7. Season
8. Package code
9. Name of the customer
10. Contact number of the customer
11. Email address of the customer

Fixed input:-

1. Number of people per room
2. Maximum people for a vacation package
3. Minimum people for a vacation package
4. GST

**Outputs:-**

1. Region of vacation
2. Season
3. Package code
4. Package type
5. Number of people per room
6. Vacation type
7. Duration of vacation
8. Number of adult
9. Number of children
10. Number of room needed
11. Name of the customer
12. Contact number of the customer
13. Email address of the customer
14. Total basic price of the package
15. Final price of the package

**Process:-**

The question wants us to help customers of the company, Dream Vacation Company to choose their desired and preferred vacation based on a few of aspects.

Customers can either choose:

-The region they want to have their vacation such as Europe, Central Asia, South East Asia and East Asia.

-The duration or time of the vacation such as 5 days, 7 days and 12 days.

-The type of vacation that they want to have such as relaxing & adventure and sightseeing & shopping.

-The package type of the vacation they enjoy such as luxury, comfort and normal.

-The season of the destination they want to have such as spring, summer, autumn and winter.

-The package code based on the region that customers want.

After that, customers are required to enter their name, contact number and email address to the program in order to record their preferred package for the invoice.

The selections from customers will be passed back to respective class member.

Then, for the calculation for the total basic price, it is based on the number of the adult and children that are entered by the customer. The following formulae are used based on the region that the customer has chosen.

For Europe: basicprice = (adult \* 3500) + (children \* 2500)

For Central Asia: basicprice = (adult \* 4800) + (children \* 3700)

For South East Asia: basicprice = (adult \* 2500) + (children \* 2000)

For East Asia: basicprice = (adult \* 4500) + (children \* 3500)

If the number of adult is more than or equal to 20, discount is given to the customer and the total basic price is recalculated.

discount = basicprice \* 0.08

basicprice = basicprice – discount

The total price is calculated based on the basic price and also the extra charges during the package selection.

If customer choose Relaxing & Adventure as the vacation type, the price will be calculated by using the formula:

totalprice = totalprice + (basicprice \* 0.05)

And there is no extra charge for Sightseeing & Shopping.

If customer choose the first package type which is Luxury, the price will be calculated by using the formula:

totalprice = totalprice + (basicprice \* 0.35)

else if customer choose the second package type which is Comfort, the price will be calculated by using the formula:

totalprice = totalprice + (basicprice \* 0.15)

else if customer choose the third package type which is Normal, there is no extra charge.

For the season which are Spring and Autumn, there is an extra charge and it is calculated by:

totalprice = totalprice + (basicprice \* 0.07)

For Summer: totalprice = totalprice + (basicprice \* 0.10)

And for Wintertotalprice = totalprice - (basicprice \* 0.18)

If the customer choose 7 days as the duration, the price will be calculated based on the package type the customer has chosen:

For Luxury: totalprice = totalprice + (1500 \* (adult + children))

For Comfort: totalprice = totalprice + (1000 \* (adult + children))

For Normal: totalprice = totalprice + (750 \* (adult + children))

Else if the customer choose 12 days as the duration, the price will be calculated based on the package type the customer has chosen:

For Luxury: totalprice = totalprice + (3500 \* (adult + children))

For Comfort: totalprice = totalprice + (1800 \* (adult + children))

For Normal: totalprice = totalprice + (1200 \* (adult + children))

And there is no extra charge for 5 days duration.

Number of room needed is calculated by using the formula:

roomneeded = (adult + children) / peopleperroom

and if the remainder of the total number of adult and children divides by number of people per room not equals to 0, the number of room needed will be incremented by 1.

Then, 6% of GST will be included in the total price of the package by the formula:

totalprice = totalprice \* 1.06

Lastly, the program will show all of the information to the customer in order to confirm the personal information and the preferred package that is chosen by the customer.

**Constraints of the program:-**

1. For luxury package type, minimum number of people must be 5, maximum number of people must be 15.
2. For comfort package type, minimum number of people must be 10, maximum number of people must be 20.
3. For normal package type, minimum number of people must be 20, maximum number of people must be 30.
4. For luxury package type, maximum 2 people per room.
5. For comfort package type, maximum 3 people per room.
6. For normal package type, maximum 4 people per room.

**Design of the Program:**

**UML diagram and inheritance/composition hierarchy**

|  |
| --- |
| BasicVac |
| #adult: int  #children: int  #region: int  #duration: int  #basicprice: double  #discount: double |
| +CalBasicPrice(): void  +ShowBasicPrice(): void  +GetBasicPrice(): double |

|  |
| --- |
| DetailsVac |
| <<friend>> input(DetailsVac&): void |
| -totalprice: double  -roomneeded: int  -vacationtype: int  -packagetype: int  -season: int  -packagecode: int  -peopleperroom: int  -contact: ContactPerson  -maxpeople: int  -minpeople: int |
| +calculateDetails(): void  +showDetails(): void  +passname(string): void  +passcontactno(string): void  +passemail(string): void |

|  |
| --- |
| ContactPerson |
| -name: string  -contactno: string  -email: string |
| +receivename(string): void  +receivecontactno(string): void  +receiveemail(string): void  +getname(): string  +getcontactno(): string  +getemail(): string |

**Class Description**

**Class BasicVac**

Purpose: To calculate the total basic price of the vacation package selected by the customer

Variables:-

1. adult – to store the number of adult
2. children – to store the number of children
3. region – to store the region selected by the customer
4. duration – to store the duration selected by the customer
5. basicprice – to store the total basic price calculated
6. discount – to store the discount calculated in order to calculate the new total basic price

Functions:-

1. CalBasicPrice() – void function to calculate the total basic price of the package that customer has chosen
2. ShowBasicPrice() – void function to display the total basic price of the package that customer has chosen
3. GetBasicPrice() – it is a integer return type function that returns the total basic price

**Class DetailsVac**

Purpose: To receive all the values that are inputted in the friend function, and to calculate the total price of the package chosen by the customer depending on the criteria that the customer chose.

Variables:-

1. totalprice – to receive the total basic price from class BasicVac then used to calculate the total price of the package
2. roomneeded – to calculate the total number of room needed by the customer
3. vacationtype – to store the vacation type selected by the customer
4. packagetype – to store the package type selected by the customer
5. season – to store the preferred season selected by the customer
6. packagecode – to store the package code selected by the customer
7. peopleperroom – to indicate the number of people allowed per room depending on the package type selected by the customer
8. contact – this is a composition to class ContactPerson
9. maxpeople – to store the maximum people allowed in the vacation package
10. minpeople – to store the minimum people allowed in the vacation package

Functions:-

1. calculateDetails() – void function to calculate the total price of the package chosen by the customer
2. showDetails() – void function to show the details of the package chosen by the customer
3. passname(string) – void function to pass the name of the customer which is received from the friend function to class ContactPerson
4. passcontactno(string) – void function to pass the contact number of the customer which is received from the friend function to class ContactPerson
5. passemail(string) – void function to pass the email address of the customer which is received from the friend function to class ContactPerson

**Selected function:-**

validation(int min, int max)

* this is a integer return type function which takes 2 parameter which are the minimum and maximum values of the selection for each question prompted respectively
* this function will take input and validate the input before passing back to the variable assigned.

validation2(int mode)

* this is a string return type function which takes 1 parameter which is the mode of the validation
* each mode will take input from customer first
* if the parameter is 1, then validation of name will be executed
* if the parameter received is 2, then validation of contact number will be executed
* if the parameter received is 3, then validation of email address will be executed
* after validation is completed, the function will return the string type input to the variable assigned

**C++ code (source code)**

**132893.cpp (main.cpp)**

1. #include <iostream>
2. #include <iomanip>
3. #include <cstdlib>
4. #include <windows.h>
5. #include "DetailsVac.h"
6. #include <string>
8. **using** **namespace** std;
10. **int** validation(**int**, **int**);
11. string validation2(**int**);
12. **void** welcome();
13. **void** loading();
14. **void** title();
15. **int** main()
16. {
17. **char** proceed = 'Y';
18. DetailsVac details1; //Definining object of DetailsVac
19. cout << fixed << showpoint << setprecision(2);
20. system("color 8F");
21. **while**(proceed == 'Y' || proceed == 'y')
22. {
23. system("cls");
24. welcome();
25. title();
26. Sleep(1500);
27. system("cls");
28. cout << endl << endl << endl << endl << endl << endl << endl << endl <<  "\t\t\t\t\t\tLet's get started." << endl;
29. Sleep(800);
30. input(details1); //friend function is being used to receive inputs from user
31. details1.CalBasicPrice();
32. details1.ShowBasicPrice();
33. loading();
34. details1.calculateDetails();
35. details1.showDetails();
36. cout << "Do you want to try again? Y or y for yes, any key for exit." << endl << "Please enter your choice: ";
37. cin >> proceed;
38. }
39. system("pause");
40. **return** 0;
41. }
43. **int** validation(**int** min, **int** max) //Validation function for all the inputs for integer value, minimum and maximum value are passed in as the range of selection
44. {
45. **int** input;
46. cin >> input;
47. **while**(input < min || input > max || !cin || cin.get() != '\n')
48. {
49. cin.clear();
50. cin.ignore(100, '\n');
51. cout << endl << endl << "\t\t\tInvalid input. Please try again: ";
52. cin >> input;
53. }
54. **return** input;
55. }
57. string validation2(**int** mode) //Validation function for name, contact number and email address
58. {
59. string input;
60. **bool** isemail = **false**;
61. **switch**(mode) //3 modes for 3 types of validation, depending on the parameter received
62. {
63. **case** 1: getline(cin, input);
64. **while**(input.find\_first\_of("`1234567890-=/\*+~!@#$%^&\*()[]{}|;'"":/.,<>?") != string::npos)
65. {
66. cin.clear();
67. cin.ignore(100, '\n');
68. cout << endl << endl << "\t\t\tInvalid input! Please enter your name again: ";
69. getline(cin, input);
70. }
71. **break**;
72. **case** 2: getline(cin, input);
73. **while**(input.length() < 10 || input.length() > 11 || input[0] != '0' || (input.find\_first\_of("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`-=/\*+~!@#$%^&\*() []{}|;'" ":/.,<>?") != string::npos))
74. {
75. cin.clear();
76. cin.ignore(100, '\n');
77. cout << endl << endl << "\t\t\tInvalid input. Please enter your phone number again: ";
78. getline(cin, input);
79. }
80. **break**;
81. **case** 3: getline(cin, input);
82. **while**(!isemail)
83. {
84. **if**(input.find(".com") != string::npos && input.find("@") != string::npos) //Validation for .com
85. isemail = **true**;
86. **else**
87. {
88. cin.clear();
89. cin.ignore(100, '\n');
90. cout << endl << endl << "\t\t\tInvalid input. Please enter your email address again: ";
91. getline(cin, input);
92. }
93. }
94. **while**(input.find\_first\_of("`~!#$%^&\*()\_+-=,< >?/;:'" "'[]{}\|") != string::npos)
95. {
96. cin.clear();
97. cin.ignore(100, '\n');
98. cout << endl << endl << "\t\t\tInvalid input. Please enter your email address again: ";
99. getline(cin, input);
100. }
101. **break**;
102. }
103. **return** input;
104. }
106. **void** loading() //function for the loading screen
107. {
108. **char** c = '\_';
109. **for**(**int** i = 0; i <=100; i++)
110. {
111. system("cls");
112. cout << endl << endl << endl << endl << endl;
113. cout << "\t\t\t\t\t\tLoading... " << i << "%" << endl << endl << endl;
114. **for**(**int** j = 0; j < i; j++)
115. {
116. cout << c;
117. }
118. Sleep(1);
119. }
120. }
122. **void** welcome() //function for the welcome screen
123. {
124. cout << endl << endl << endl << endl << endl << "\t\t\t\t\t\t";
125. **char** c[13] = {'W', 'e', 'l', 'c', 'o', 'm', 'e', ' ', 't', 'o', '.', '.', '.'};
126. **for**(**int** i = 0; i < 13; i++)
127. {
128. cout << c[i];
129. Sleep(100);
130. }
131. }
133. **void** title() //Introduction for the program
134. {
135. **for**(**int** i = 0; i < 3; i++)
136. {
137. system("cls");
138. Sleep(400);
139. cout << endl << endl << endl << endl << endl << "\t\t\t\t\t\tWelcome to...";
140. cout << endl << endl << endl;
141. cout << " \t\t\t\tMyDreamV - Your Personal Vacation Cost Estimator" << endl << endl;
142. cout << "\t\t\t\tWe can help you to estimate the cost for your vacation " << endl;
143. cout << "\t\t\t\t\t\tbased on your choice." << endl << endl << endl;
144. cout << "\t\t\t\t\tBy Dream Vacation Company";
145. Sleep(400);
146. }
147. }
148. **void** input(DetailsVac& d) //friend function for receiving all the inputs from user
149. {
150. string name, contactno, email;
151. system("cls");
152. cout << endl << endl << endl << "\t\t\tPlease select the region as the number mentioned below: " << endl << endl;
153. cout << "\t\t\t1. Europe" << endl << "\t\t\t2. Central Asia" << endl << "\t\t\t3. South East Asia" << endl << "\t\t\t4. East Asia" << endl << endl;
154. cout << "\t\t\tPlease enter your choice: ";
155. d.region = validation(1, 4); //Parameters for validation are minimum and maximum values allowed for the selection
156. system("cls");
157. **if**(d.region == 1 || d.region == 2 || d.region == 4) //Only Europe, Central Asia and East Asia have selection for season
158. {
159. cout << endl << endl << endl << "\t\t\tPlease select the season that you want to have during your vacation: " << endl << endl;
160. cout << "\t\t\t1. Spring" << endl << "\t\t\t2. Summer" << endl << "\t\t\t3. Autumn" << endl << "\t\t\t4. Winter" << endl << endl;
161. cout << "\t\t\tPlease enter your choice: ";
162. d.season = validation(1, 4);
163. }
164. **else** **if**(d.region == 3)
165. d.season = 0; //South East Asia countries do not have season
166. system("cls");
167. **if**(d.region == 1) //Package code will be shown to user by referring to chosen region respectively
168. {
169. cout << endl << endl << endl << "\t\t\tPlease type the package code as listed below: " << endl << endl;
170. cout << "\t\t\t1 - EU.1 Paris, Brussel, Amsterdam" << endl << "\t\t\t2 - EU.2 Paris, Strasbourg, Luxemburg" << endl << endl;
171. cout << "\t\t\tPlease enter your choice: ";
172. d.packagecode = validation(1, 2);
173. }
174. **if**(d.region == 2)
175. {
176. cout << endl << endl << endl << "\t\t\tPlease type the package code as listed below: " << endl << endl;
177. cout << "\t\t\t1 - CA.1 Samarkand, Bukhara, Tashkent" << endl << endl;
178. cout << "\t\t\tPlease enter your choice: ";
179. d.packagecode = validation(1,1);
180. }
181. **if**(d.region == 3)
182. {
183. cout << endl << endl << endl << "\t\t\tPlease type the package code as listed below: " << endl << endl;
184. cout << "\t\t\t1 - SEA.1 Kuala Lumpur, Penang, Langkawi" << endl << "\t\t\t2 - SEA.2 Kuala Lumpur, Kuching, Kota Kinabalu" << endl << endl;
185. cout << "\t\t\tPlease enter your choice: ";
186. d.packagecode = validation(1, 2);
187. }
188. **if**(d.region == 4)
189. {
190. cout << endl << endl << endl << "\t\t\tPlease type the package code as listed below: " << endl << endl;
191. cout << "\t\t\t1 - EA,1 Beijing, Shanghai, Hangzhou" << endl << "\t\t\t2 - EA.2 Seoul, Sokcho" << endl << "\t\t\t3 - EA.3 Seoul, Busan, Jeju" << endl << "\t\t\t4 - EA.4 Tokyo, Osaka" << endl << endl;
192. cout << "\t\t\tPlease enter your choice: ";
193. d.packagecode = (validation(1, 4));
194. }
195. system("cls");
196. cout << endl << endl << endl << "\t\t\tPlease select the perferred package type that you want: " << endl << endl;
197. cout << "\t\t\t1. Luxury" << endl << "\t\t\t\t-35% extra charge on total basic price of the package" << endl << "\t\t\t\t-Stay in 5 star hotel, maximum 2 people in a room"<< endl << "\t\t\t\t-Extra 2 meals are provided" << endl << "\t\t\t\t-Minumum 5 people, Maximum 15 people" << endl << endl <<
198. "\t\t\t2. Comfort" << endl << "\t\t\t\t-15% extra charge on total basic price of the package" << endl << "\t\t\t\t-Stay in 3/4 star hotel, maximum 3 people in a room" << endl << "\t\t\t\t-Minimum 10 people, Maximum 20 people" << endl <<
199. "\t\t\t3. Normal" << "\t\t\t\t-No extra charge" << endl << "\t\t\t\t-Stay in 3 star hotel, 3-4 people in a room" << endl << "\t\t\t\t-Minimum 20 people, maximum 30 people" << endl << endl;
200. cout << "\t\t\tPlease enter your choice: ";
201. d.packagetype = validation(1, 3);
203. **if**(d.packagetype == 1) //For luxury package type
204. {
205. d.peopleperroom = 2;
206. d.minpeople = 5;
207. d.maxpeople = 15;
208. }
209. **else** **if**(d.packagetype == 2) //For comfort pacakge type
210. {
211. d.peopleperroom = 3;
212. d.minpeople = 10;
213. d.maxpeople = 20;
214. }
215. **else** **if**(d.packagetype == 3) //For normal package type
216. {
217. d.peopleperroom = 4;
218. d.minpeople = 20;
219. d.maxpeople = 30;
220. }
221. system("cls");
222. cout << endl << endl << endl << "\t\t\tPlease select the type of vacation as the number mentioned below: " << endl << endl;
223. cout << "\t\t\t1. Sightseeing & Shopping" << endl << "\t\t\t2. Relaxing & Adventure" << endl << endl;
224. cout << "\t\t\tPlease enter your choice: ";
225. d.vacationtype = validation(1, 2);
226. system("cls");
227. cout << endl << endl << endl << "\t\t\tPlease select the duration of vacation that you want: " << endl << endl;
228. cout << "\t\t\t1. 5 days - No extra charge" << endl;
229. cout << "\t\t\t2. 7 days - Extra charge for each person: Luxury - RM1500, Comfort - RM1000, Normal - RM750" << endl;
230. cout << "\t\t\t3. 12 days - Extra charge for each person: Luxury - RM3500, Comfort - RM1800, Normal - RM1200" << endl << endl;
231. cout << "\t\t\tPlease enter your choice: ";
232. d.duration = validation(1, 3);
233. system("cls");
234. cout << endl << endl << endl << "\t\t\tPlease enter the number of adult: ";
235. d.adult = validation(0, 30);
236. cout << endl << endl;
237. cout << endl << endl << "\t\t\tPlease enter the number of children: ";
238. d.children = validation(0, 30);
239. **while**((d.adult + d.children) < d.minpeople || (d.adult + d.children) > d.maxpeople) //While loop for detecing the total number of people involved in the vacation package
240. {
241. system("cls");
242. cout << endl << endl << endl << "\t\t\tInvalid total number of people. Total number of people must be at least " << d.minpeople << " and at most " << d.maxpeople << "." << endl << "\t\t\tTry again. " << endl << endl;
243. cout << "\t\t\tPlease enter the number of adult: ";
244. d.adult = validation(0, 30);
245. cout << endl << endl;
246. cout << "\t\t\tPlease enter the number of children: ";
247. d.children = validation(0, 30);
248. }
250. system("cls");
251. cout << endl << endl << endl << "\t\t\tPlease enter your name: ";
252. name = validation2(1);
253. d.passname(name);
255. cout << endl << endl << "\t\t\tPlease enter your contact number: ";
256. contactno = validation2(2);
257. d.passcontactno(contactno);
259. cout << endl << endl << "\t\t\tPlease enter your email address: ";
260. email = validation2(3);
261. d.passemail(email);
262. }

**BasicVac.h**

1. #ifndef BASICVAC\_H
2. #define BASICVAC\_H
4. **class** BasicVac
5. {
6. **public**:
7. **void** CalBasicPrice();
8. **void** ShowBasicPrice();
9. **double** GetBasicPrice();
10. BasicVac();
11. ~BasicVac();
12. **protected**:
13. **int** adult;
14. **int** children;
15. **int** region;
16. **int** duration;
17. **double** basicprice;
18. **double** discount;
19. };
21. #endif

**BasicVac.cpp**

1. #include <iostream>
2. #include "BasicVac.h"
4. **using** **namespace** std;
6. BasicVac::BasicVac()
7. {
8. adult = 0, children = 0, region = 0, duration = 0;
9. }
11. BasicVac::~BasicVac()
12. {
13. adult = 0, children = 0, region = 0, duration = 0;
14. }
16. **void** BasicVac::CalBasicPrice() //Calculation for the total basic price of the vacation package
17. {
18. **if**(region == 1)
19. basicprice = (adult \* 3500) + (children \* 2500);
20. **else** **if**(region == 2)
21. basicprice = (adult \* 4800) + (children \* 3700);
22. **else** **if**(region == 3)
23. basicprice = (adult \* 2500) + (children \* 2000);
24. **else** **if** (region == 4)
25. basicprice = (adult \* 4500) + (children \* 3500);
27. **if**(adult >= 20) //Discount is given if number of adults is more than or equals to 20
28. {
29. discount = basicprice \* 0.08;
30. basicprice = basicprice - discount;
31. }
32. }
34. **void** BasicVac::ShowBasicPrice() //Total basic price will be displayed
35. {
36. cout << "\t\tTotal basic price of the package is: RM" << basicprice << endl;
37. }
39. **double** BasicVac::GetBasicPrice()
40. {
41. **return** basicprice;
42. }

**DetailsVac.h**

1. #ifndef DETAILSVAC\_H
2. #define DETAILSVAC\_H
4. #include "BasicVac.h"
5. #include "ContactPerson.h"
7. **using** **namespace** std;
9. **class** DetailsVac : **public** BasicVac
10. {
11. **friend** **void** input(DetailsVac&);
12. **public**:
13. **void** calculateDetails();
14. **void** showDetails();
15. **void** passname(string);
16. **void** passcontactno(string);
17. **void** passemail(string);
18. DetailsVac();
19. ~DetailsVac();
20. **private**:
21. **double** totalprice;
22. **int** roomneeded;
23. **int** vacationtype;
24. **int** packagetype;
25. **int** season;
26. **int** packagecode;
27. **int** peopleperroom;
28. ContactPerson contact;
29. **int** maxpeople;
30. **int** minpeople;
31. };
33. #endif

**DetailsVac.cpp**

1. #include <iostream>
2. #include "DetailsVac.h"
4. **using** **namespace** std;
6. DetailsVac::DetailsVac()
7. {
8. totalprice = 0, roomneeded = 0, vacationtype = 0, packagetype = 0, season = 0;
9. }
11. DetailsVac::~DetailsVac()
12. {
13. totalprice = 0, roomneeded = 0, vacationtype = 0, packagetype = 0, season = 0;
14. }
16. **void** DetailsVac::passname(string name) //Function to pass the name of user from friend function to ContactPerson
17. {
18. contact.receivename(name);
19. }
21. **void** DetailsVac::passcontactno(string contactno) //Function to pass the contact number of user from friend function to ContactPerson
22. {
23. contact.receivecontactno(contactno);
24. }
26. **void** DetailsVac::passemail(string email) //Function to pass the email address of the user from friend function to ContactPerson
27. {
28. contact.receiveemail(email);
29. }
31. **void** DetailsVac::calculateDetails() //Calculation for the total price of the vacation package
32. {
33. totalprice = GetBasicPrice();
34. **if**(vacationtype == 2)
35. totalprice = totalprice + (basicprice \* 0.05); //Extra charge for Relaxing & Adventure
37. **if**(packagetype == 1) //Extra charge for luxury and comfort package type
38. totalprice = totalprice + (basicprice \* 0.35);
39. **else** **if**(packagetype == 2)
40. totalprice = totalprice + (basicprice \* 0.15);
42. **if**(region == 1 || region == 2 || region == 4)
43. {
44. **if**(season == 1 || season == 3)
45. totalprice = totalprice + (basicprice \* 0.07); //Extra charge for spring and autumn
46. **else** **if**(season == 2)
47. totalprice = totalprice + (basicprice \* 0.10); //Extra charge for summer
48. **else** **if**(season == 4)
49. totalprice = totalprice - (basicprice \* 0.18); //For Winter, discount is given
50. }
52. **if**(duration == 2) //Extra charge for 7 days vacation packages
53. {
54. **if**(packagetype == 1)
55. totalprice = totalprice + (1500 \* (adult + children));
56. **else** **if**(packagetype == 2)
57. totalprice = totalprice + (1000 \* (adult + children));
58. **else** **if**(packagetype == 3)
59. totalprice = totalprice + (750 \* (adult + children));
60. }
61. **else** **if**(duration == 3) //Extra charge for 12 days vacation packages
62. {
63. **if**(packagetype == 1)
64. totalprice = totalprice + (3500 \* (adult + children));
65. **else** **if**(packagetype == 2)
66. totalprice = totalprice + (1800 \* (adult + children));
67. **else** **if**(packagetype == 3)
68. totalprice = totalprice + (1200 \* (adult + children));
69. }
71. roomneeded = (adult + children) / peopleperroom; //Formula to calculate number of room needed
72. **if**((adult + children) % peopleperroom != 0) //If remaining number of people is not same as the maximum number of people per room
73. roomneeded++;
75. totalprice = totalprice \* 1.06; //GST is included in totalprice
76. }
78. **void** DetailsVac::showDetails() //Print all of the information of the chosen package by user
79. {
80. system("cls");
81. cout << endl << endl << "\t\t----------Final Information----------" << endl << endl;
82. cout << "\t\tRegion: ";
83. **if**(region == 1)
84. cout << "Europe" << endl;
85. **else** **if**(region == 2)
86. cout << "Central Asia" << endl;
87. **else** **if**(region == 3)
88. cout << "South East Asia" << endl;
89. **else** **if**(region == 4)
90. cout << "East Asia" << endl;
92. cout << "\t\tSeason: ";
93. **if**(season == 0)
94. cout << "No season is available" << endl;
95. **else** **if**(season == 1)
96. cout << "Spring" << endl;
97. **else** **if**(season == 2)
98. cout << "Summer" << endl;
99. **else** **if**(season == 3)
100. cout << "Autumn" << endl;
101. **else** **if**(season == 4)
102. cout << "Winter" << endl;
104. cout << "\t\tPackage code: ";
105. **if**(region == 1)
106. {
107. **if**(packagecode == 1)
108. cout << "EU.1 Paris, Brussel, Amsterdam" << endl;
109. **else** **if**(packagecode == 2)
110. cout << "EU.2 Paris, Strasbourg, Luxemburg" << endl;
111. }
112. **else** **if**(region == 2)
113. cout << "CA.1 Samarkand, Bukhara, Tashkent" << endl;
114. **else** **if**(region == 3)
115. {
116. **if**(packagecode == 1)
117. cout << "SEA.1 Kuala Lumpur, Penang, Langkawi" << endl;
118. **else** **if**(packagecode == 2)
119. cout << "SEA.2 Kuala Lumpur, Kuching, Kota Kinabalu" << endl;
120. }
121. **else** **if**(region == 4)
122. {
123. **if**(packagecode == 1)
124. cout << "EA.1 Beijing, Shanghai, Hangzhou" << endl;
125. **else** **if**(packagecode == 2)
126. cout << "EA.2 Seoul, Sokcho" << endl;
127. **else** **if**(packagecode == 3)
128. cout << "EA.3 Seoul, Busan, Jeju" << endl;
129. **else** **if**(packagecode == 4)
130. cout << "EA.4 Tokyo, Osaka" << endl;
131. }
133. cout << "\t\tPreferred package type: ";
134. **if**(packagetype == 1)
135. cout << "Luxury" << endl;
136. **else** **if**(packagetype == 2)
137. cout << "Comfort" << endl;
138. **else** **if**(packagetype == 3)
139. cout << "Normal" << endl;
141. cout << "\t\tNumber of people per room: ";
142. **if**(peopleperroom == 2)
143. cout << "2 people" << endl;
144. **else** **if**(peopleperroom == 3)
145. cout << "3 people" << endl;
146. **else** **if**(peopleperroom == 4)
147. cout << "4 people" << endl;
149. cout << "\t\tType of vacation: ";
150. **if**(vacationtype == 1)
151. cout << "Sightseeing & Shopping" << endl;
152. **else** **if**(vacationtype == 2)
153. cout << "Relaxing & Adventure" << endl;
155. cout << "\t\tDuration of vacation: ";
156. **if**(duration == 1)
157. cout << "5 days" << endl;
158. **else** **if**(duration == 2)
159. cout << "7 days" << endl;
160. **else** **if**(duration == 3)
161. cout << "12 days" << endl;
163. cout << "\t\tNumber of adult: " << adult << endl;
164. cout << "\t\tNumber of children: " << children << endl;
166. cout << "\t\tNumber of room: " << roomneeded << endl << endl;
168. cout << "\t\tYour name: " << contact.getname() << endl;
169. cout << "\t\tYour contact number: " << contact.getcontactno() << endl;
170. cout << "\t\tYour email address: " << contact.getemail() << endl << endl;
172. ShowBasicPrice();
173. cout << "\t\tThe final price is: RM" << totalprice << endl << endl;
174. cout << "\t\tHave a nice day :)" << endl << endl;
175. }

**ContactPerson.h**

1. #ifndef CONTACTPERSON\_H
2. #define CONTACTPERSON\_H
3. #include <string>
5. **using** **namespace** std;
7. **class** ContactPerson
8. {
9. **public**:
10. **void** receivename(string);
11. **void** receivecontactno(string);
12. **void** receiveemail(string);
13. string getname();
14. string getcontactno();
15. string getemail();
16. ContactPerson();
17. ~ContactPerson();
18. **private**:
19. string name;
20. string contactno;
21. string email;
22. };
24. #endif

**ContactPerson.cpp**

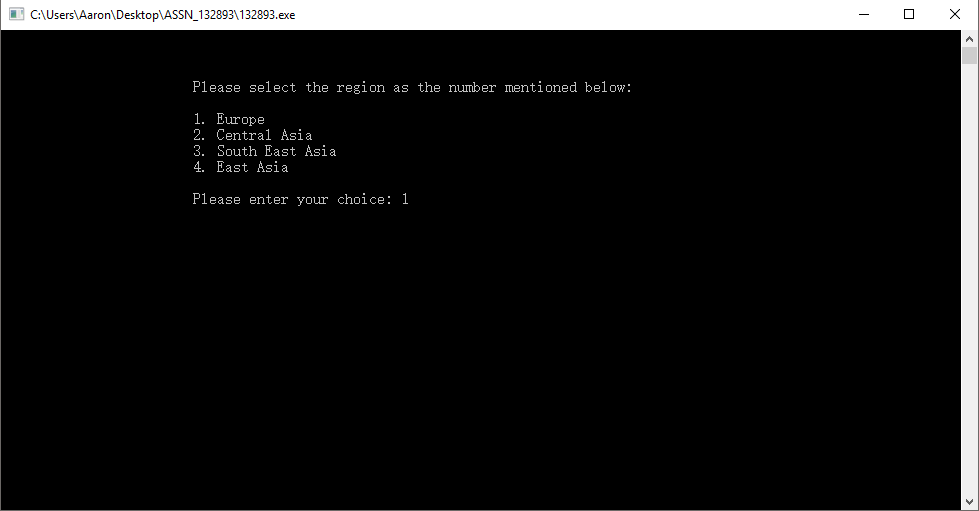
1. #include <iostream>
2. #include "ContactPerson.h"
4. ContactPerson::ContactPerson()
5. {
6. name = " "; contactno = " "; email = " ";
7. }
9. ContactPerson::~ContactPerson()
10. {
11. name = " "; contactno = " "; email = " ";
12. }
14. **void** ContactPerson::receivename(string n) //Function which is used to receive name of the user
15. {
16. name = n;
17. }
19. **void** ContactPerson::receivecontactno(string c) //Function which is used to receive contact number of the user
20. {
21. contactno = c;
22. }
24. **void** ContactPerson::receiveemail(string e) //Function which is used to receive email address of the user
25. {
26. email = e;
27. }
29. string ContactPerson::getname() //Function used to return name of the user
30. {
31. **return** name;
32. }
34. string ContactPerson::getcontactno() //Function used to return contact number of the user
35. {
36. **return** contactno;
37. }
39. string ContactPerson::getemail() //Function used to return email address of the user
40. {
41. **return** email;
42. }

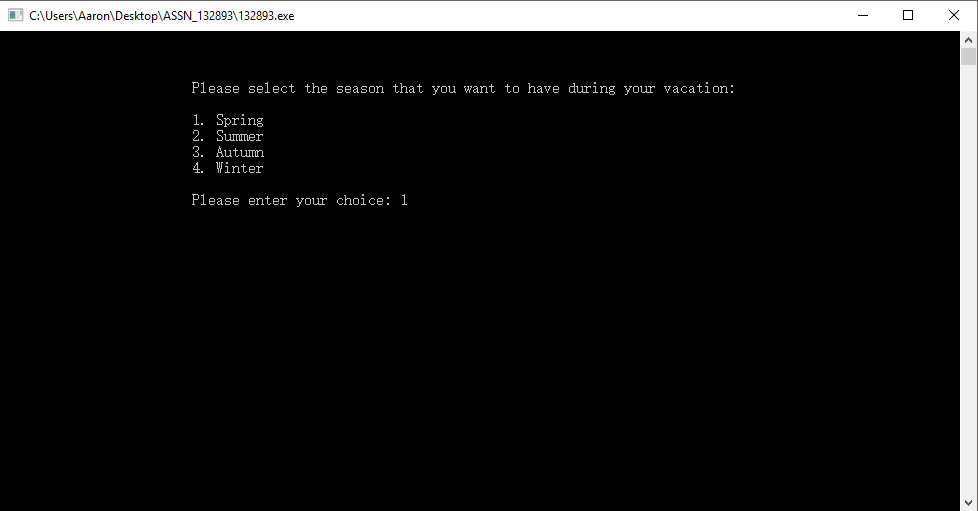
**Test Data**

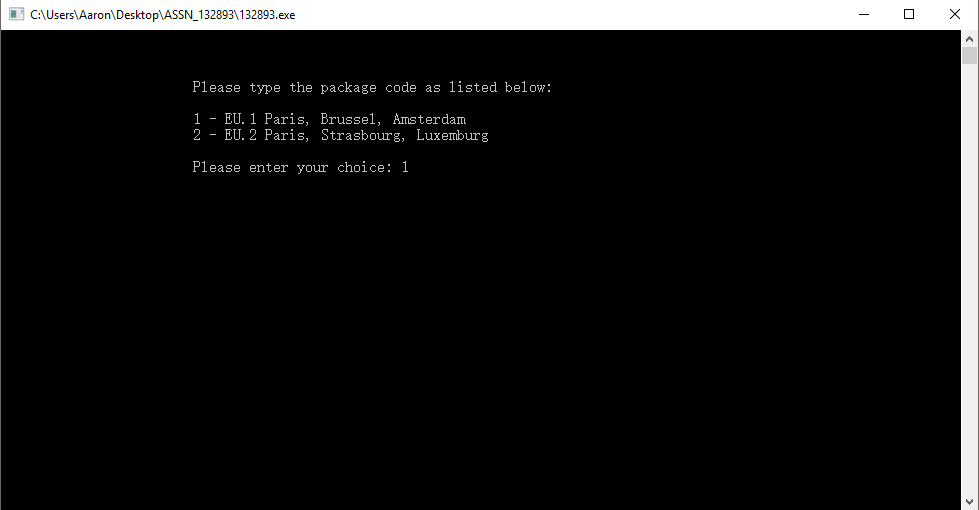
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Choice of region | Choice of season | Choice of package code | Choice of package type | Choice of vacation type | Choice of duration | No. of adult | No. of children | Name | Contact Number | Email Address | Total basic price (RM) | Total price after GST (RM) |
| 1 | 1 | 1 | 1 | 1 | 1 | 10 | 5 | Aaron | 0162345667 | aaron@gmail.com | 47500.00 | 71497.00 |
| 2 | 2 | 4 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |
| 4 | 1 | 2 | 1 | 2 | 8 | 4 | Chew | 0165366273 | chew,123@gmail.com | Inv. | Inv. |
| 2 | 3 | 1 | 2 | 3 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |
| 5 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |
| 1 | 1 | 3 | 1 | 2 | 10 | 5 | Chong | contact | Inv. | Inv. | Inv. |
| 3 | 0 | 1 | 3 | 2 | 4 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |
| 0 | 4 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |
| 4 | 4 | 1 | 2 | 1 | 1 | 4 | 10 | 12345 | Inv. | Inv. | Inv. | Inv. |
| 3 | 2 | 1 | 2 | 2 | 10 | 20 | Inv. | Inv. | Inv. | Inv. | Inv. |
| 5 | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. | Inv. |

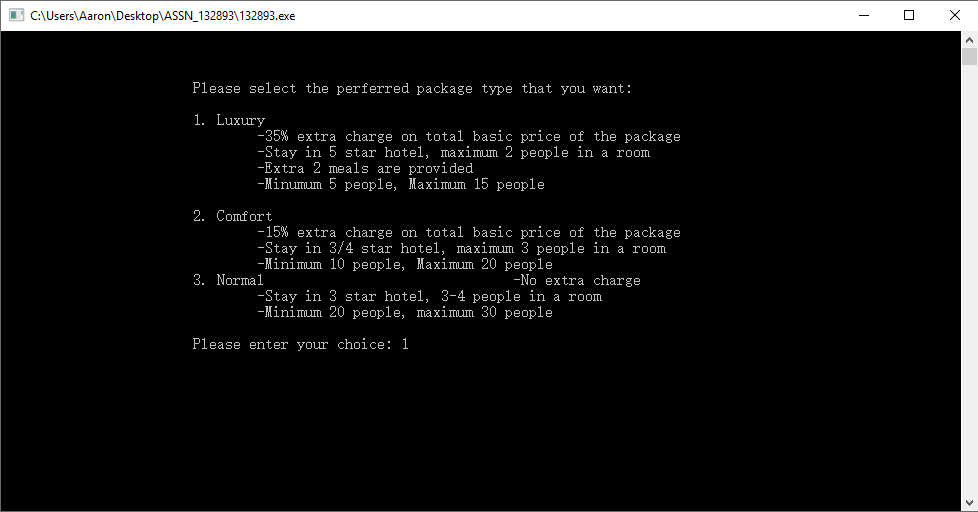
**Print screen of sample input/output**

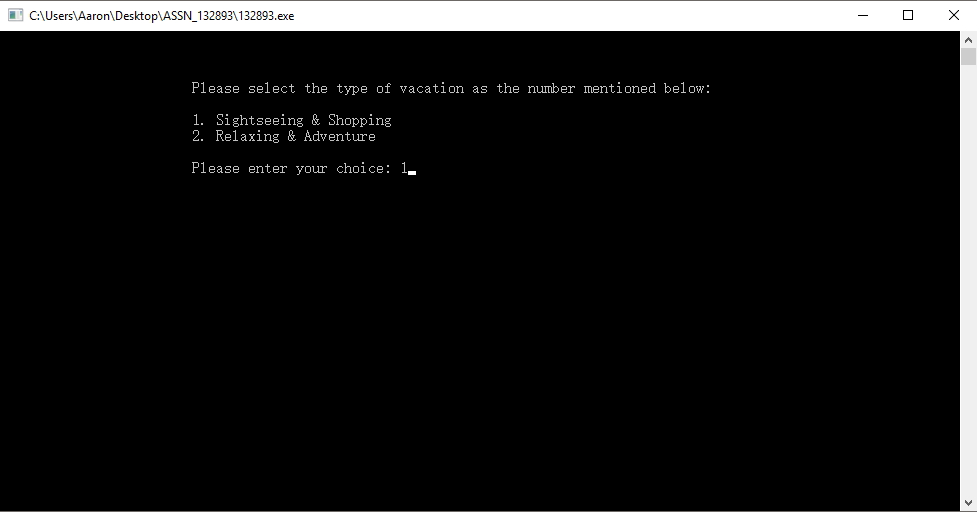
**Scenario 1: All correct inputs**

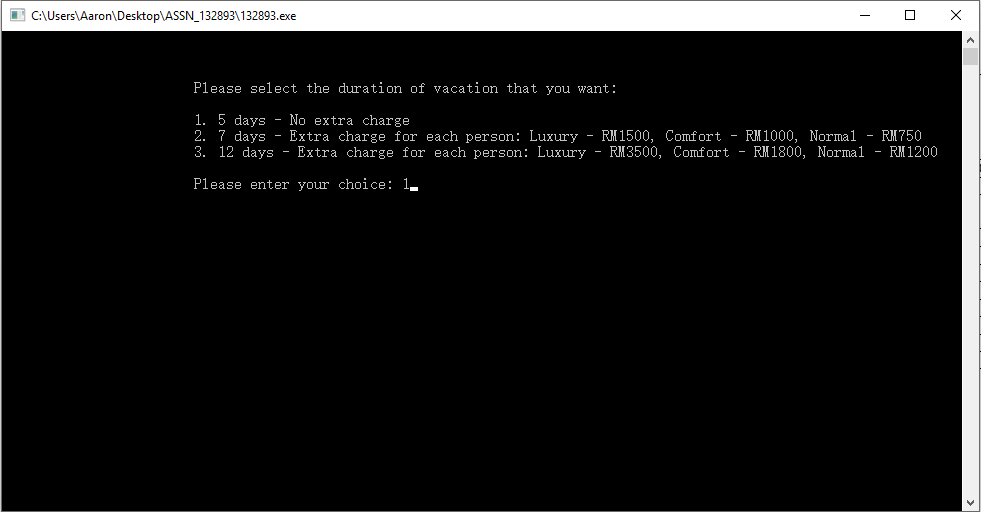


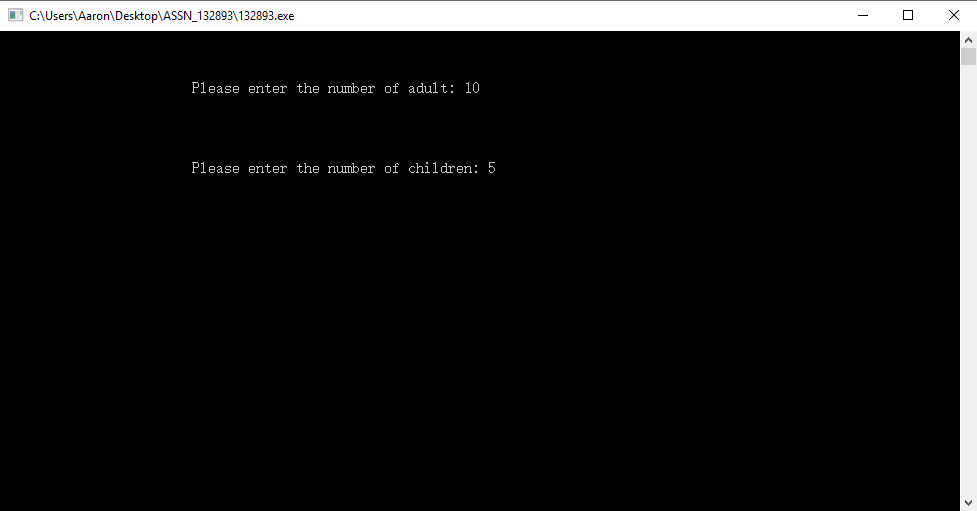


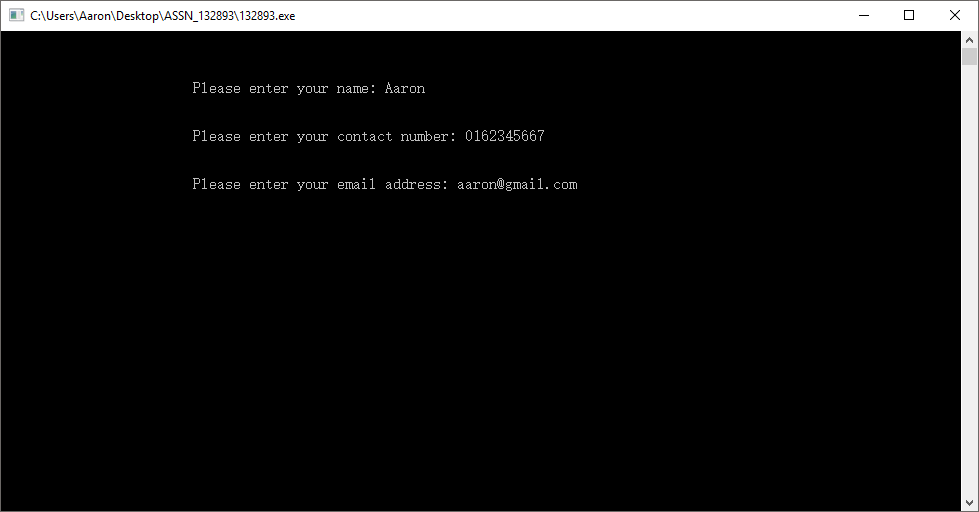


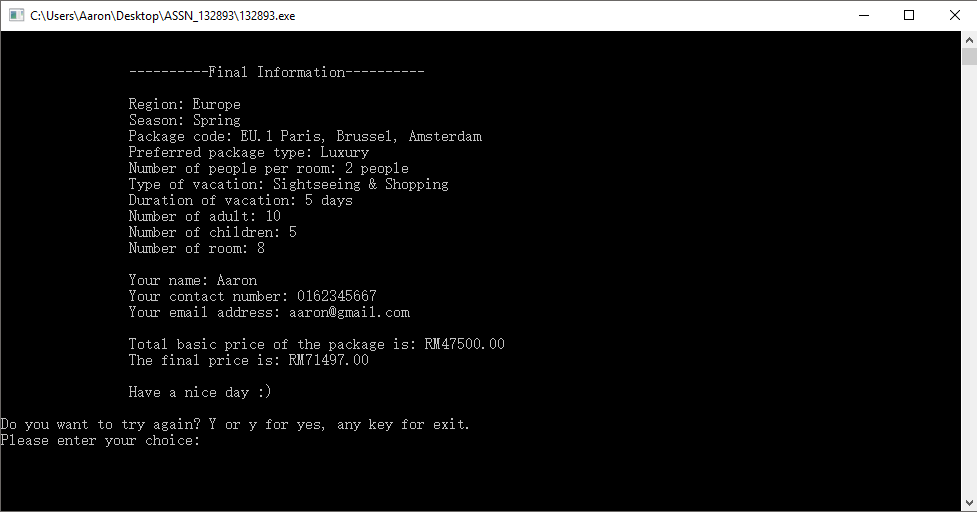












**Scenario 2: All invalid inputs**

