ComSci 31 – Winter 2019

Project 2 Report

Name – Suyash Kumar

UID – 605108040

Discussion – 1C

# Obstacles

1. Error handling (when the console doesn’t get expected input for either kind value, meal value, type value, or resident value) was somewhat tricky. Consider the case of kind value, which could take up Ticket OR BruinCard, so erroneous input would be a kind value which would be NOT Ticket AND NOT BruinCard. Similarly, for meal value, correct input would be Breakfast OR Lunch OR Dinner, so erroneous input would be NOT Breakfast AND NOT Lunch AND NOT Dinner. Error handling for other cases is similar.
2. The flow of logic was somewhat complicated, which is why I laid out sequential algorithmic steps before actually plunging into writing the code –
3. Accept input for kind value
4. Check if kind value is NOT Ticket AND NOT BruinCard, and if that’s the case terminate the program. If kind value is either Ticket OR BruinCard, proceed.
5. Check if meal value is NOT Breakfast AND NOT Lunch AND NOT Dinner, and in that case terminate the program. If meal value is either Breakfast OR Lunch OR Dinner, proceed.
6. Check if kind value is Ticket, if it is, check whether meal value is Breakfast, Lunch, or Dinner, and produce output accordingly.
7. Check if the kind value is BruinCard instead, and if it is, demand the type value, which should be either Student or Staff.
8. Check if the type value is NOT Student AND NOT Staff, and if that is the case then terminate the program. If the type value is Student OR Staff, then proceed.
9. Check if the type value is Staff, and if it is, check whether the meal type was Breakfast, Lunch, or Dinner, and produce output accordingly.
10. Check if the type value was Student instead, and demand for the resident value, which can be either Yes or No.
11. Check if the resident value is NOT Yes AND NOT No, and if that is the case, terminate the program.
12. Check if the resident value is Yes, and if it is, check if the meal value is Breakfast, Lunch, or Dinner, and produce output accordingly.
13. Check if the resident value is No instead, and if it is, check if the meal value is Breakfast, Lunch, or Dinner, and produce output accordingly.

I feel that this algorithm is robust due to its prioritization of error handling. The program terminates as soon as an error is encountered. Besides, the logic flow is quite efficient.

1. Instead of printing out a success statement in every if/ else-if construct, it is wiser to store a value in the price variable (double type) and print out everything once the logical check has been performed. It is essential to use the command cout.setf(ios::showpoint), cout.setf(ios::fixed), and cout.precision(2) to get an output with two decimal places.

# Test Cases

The test cases fall in the following two categories –

Handling good inputs –

We have the following test cases for thoroughly testing if our program is able to handle good input:

1. Breakfast with Ticket (Ticket, Breakfast)
2. Lunch with Ticket (Ticket, Lunch)
3. Dinner with Ticket (Ticket, Dinner)
4. Breakfast with BruinCard as a Staff (BruinCard, Breakfast, Staff)
5. Lunch with BruinCard as a Staff (BruinCard, Lunch, Staff)
6. Dinner with BruinCard as a Staff (BruinCard, Dinner, Staff)
7. Breakfast with BruinCard as a resident Student (BruinCard, Breakfast, Student, Yes)
8. Lunch with BruinCard as a resident Student (BruinCard, Lunch, Student, Yes)
9. Dinner with BruinCard as a resident Student (BruinCard, Dinner, Student, Yes)
10. Breakfast with BruinCard as a non-resident Student (BruinCard, Breakfast, Student, No)
11. Lunch with BruinCard as a non-resident Student (BruinCard, Lunch, Student, No)
12. Dinner with BruinCard as a non-resident Student (BruinCard, Dinner, Student, No)

Error handling –

We have the following test cases to ensure that our program is proficient in error handling (various test cases are separated by a /):

1. Erroneous input for kind value (bruin card)
2. Erroneous input for meal value (Ticket, lunch and dinner/ BruinCard, break fast)
3. Erroneous input for type value (BruinCard, Breakfast, teacher/ BruinCard, Lunch, CEO/ BruinCard, Dinner, student)
4. Erroneous input for resident value (BruinCard, Breakfast, Student, Maybe/ BruinCard, Lunch, Student, I don’t know/ BruinCard, Dinner, Student, Never mind)

All these test cases conclusively establish our programs ability to handle all combinations of good input as well as bad input.