|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testing Plan | | | | |
| Testing Age Calculation | | | | |
| Test | Inputs | Expected Output | Actual Output | Pass/Fail |
| Testing Day of Month | DOB: 2000-8-2  Today’s Date: 2019-8-1 | Age: 18.5 | Age: 18.5 | Pass |
| ‘’ | DOB: 2000-8-2 Today’s Date: 2000-8-3 | Age: 19 | Age: 19 | Pass |
| Testing Year | DOB: 1999-8-31  Today’s Date: 2019-8-31 | Age: 20 | Age: 20 | Pass |
| Testing Month | DOB: 1989-2-20  Today’s Date: 2019-1-20 | Age: 29.5 | Age: 29.5 | Pass |
| Verifying Inputs | | | | |
| Testing Days of Months | DOB: 2000-9-31 | “Enter a valid birth day.” Request input reentry | “Enter a valid birth day.” Request input reentry | Pass |
| ‘’ | DOB: 2000-8-31 | “Enter current year: “ (input accepted) | “Enter current year: “ (input accepted) | Pass |
| Testing Leap Year | DOB: 2000-2-29 | “Enter current year: “ (input accepted) | “Enter current year: “ (input accepted) | Pass |
| ‘’ | DOB: 2001-2-29 | “Enter a valid birth day.” Request input reentry | “Enter a valid birth day.” Request input reentry | Pass |
| Testing Year | Customer Birth Year: 1899 | “Enter a valid birth year. “ Request input reentry | “Enter a valid birth year. “ Request input reentry | Pass |
| Testing Negative Age | DOB: 2019-8-1  Today’s Date: 2019-2-20 | “Age: -0.5  Invalid entry.  Enter customer birth year:” | “Age: -0.5  Invalid entry.  Enter customer birth year:” | Pass |
| ‘’ | DOB: 2019-8-10  Today’s Date: 2019-8-9 | “Age: -0.5  Invalid entry.  Enter customer birth year:” | “Age: -0.5  Invalid entry.  Enter customer birth year:” | Pass |
| Testing Restrictions | | | | |
| Testing with Age 8 Customer | DOB: 2011-9-20  Today’s Date: 2019-9-23  Rating: EC | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’  (half year) | DOB: 2011-9-20  Today’s Date: 2019-9-23  Rating: E | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| ‘’  (half year) | DOB: 2011-1-2  Today’s Date: 2019-9-23  Rating: E | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2011-9-20 Today’s Date: 2019-9-23  Rating: E10 | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| ‘’ | DOB: 2011-9-20 Today’s Date: 2019-9-23  Rating: T | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| ‘’ | DOB: 2011-9-20 Today’s Date: 2019-9-23  Rating: M | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| ‘’ | DOB: 2011-9-20 Today’s Date: 2019-9-23  Rating: AO | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| Testing with Age 17 Customer | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: EC | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: E | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: E10 | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: T | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: M | “Customer can rent item.” | “Customer can rent item.” | Pass |
| ‘’ | DOB: 2002-9-23  Today’s Date: 2019-9-23  Rating: AO | “Customer cannot rent item.” | “Customer cannot rent item.” | Pass |
| Testing with Age 18 Customer | DOB: 2001-9-23  Today’s Date: 2019-9-23  Rating: AO | “Customer can rent item.” | “Customer can rent item.” | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TDD Development | | | | |
| Requirement: Correct Age Calculation | | | | |
| Test | Inputs | Expected Output | Actual Output | Pass/Fail |
| Same months | DOB: 2000-8-2  Today’s Date: 2019-8-1 | Age: 18.5 | Age: 18 | Fail |
| ‘’ | DOB: 2000-8-2  Today’s Date: 2019-8-3 | Age: 19 | Age: 19 | Pass |
| ‘’ | DOB: 2000-8-2  Today’s Date: 2019-8-2 | Age: 19 | Age: 19 | Pass |
| double ageCalculator(int birthMonth, int birthDay, int birthYear, int currentMonth, int currentDay, int currentYear)  {  double age = currentYear - birthYear - 1;  if ((birthMonth < currentMonth) || (birthMonth == currentMonth && birthDay <= currentDay)) {  age++;  }  if ((abs(currentMonth - birthMonth) >= 6 || abs(birthMonth - currentMonth) >= 6) && currentDay >= birthDay) {  age += .5;  }  return age;  } | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TDD Development | | | | |
| Requirement: Correct Age Calculation Revision | | | | |
| Test | Inputs | Expected Output | Actual Output | Pass/Fail |
| Day before birthday | DOB: 2000-12-2  Today’s Date: 2018-12-1 | Age: 17.5 | Age: 17.5 | Pass |
| Day after birthday (no half year) | DOB: 2000-8-2  Today’s Date: 2019-8-3 | Age: 19 | Age: 19.5 | Fail |
| 5 months before birthday (half year) | DOB: 2000-11-5  Today’s Date: 2019-6-8 | Age: 18.5 | Age: 18.5 | Pass |
| 10 months after birthday (half year) | DOB: 2000-1-9  Today’s Date: 2019-11-20 | Age 19.5 | Age: 19.5 | Pass |
| double ageCalculator(int birthMonth, int birthDay, int birthYear, int currentMonth, int currentDay, int currentYear)  {  double age = currentYear - birthYear - 1;  if ((birthMonth < currentMonth) || (birthMonth == currentMonth && birthDay <= currentDay)) {  age++;  }  if (birthMonth <= 6) {  if (currentMonth >= (birthMonth + 6)) {  age += .5;  }  }  else if (birthMonth > 6) {  if (currentMonth >= (birthMonth - 6)) {  age += .5;  }  }  return age;  } | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TDD Development | | | | |
| Requirement: Correct Age Calculation Revision 2 | | | | |
| Test | Inputs | Expected Output | Actual Output | Pass/Fail |
| Day before birthday | DOB: 2000-12-2  Today’s Date: 2018-12-1 | Age: 17.5 | Age: 17.5 | Pass |
| Day after birthday (no half year) | DOB: 2000-8-2  Today’s Date: 2019-8-3 | Age: 19 | Age: 19 | Pass |
| 5 months before birthday (half year) | DOB: 2000-11-5  Today’s Date: 2019-6-8 | Age: 18.5 | Age: 18.5 | Pass |
| 10 months after birthday (half year) | DOB: 2000-1-9  Today’s Date: 2019-11-20 | Age 19.5 | Age: 19.5 | Pass |
| double ageCalculator(int birthMonth, int birthDay, int birthYear, int currentMonth, int currentDay, int currentYear)  {  double age = currentYear - birthYear - 1;  if ((birthMonth < currentMonth) || (birthMonth == currentMonth && birthDay <= currentDay)) {  age++;  }  if (birthMonth <= 6) {  if (currentMonth < birthMonth) {  age += .5;  }  if (currentMonth == birthMonth && currentDay < birthDay) {  age += .5;  }  if (currentMonth >= (birthMonth + 6)) {  age += .5;  }  }  else if (birthMonth > 6) {  if (currentMonth == birthMonth && currentDay < birthDay) {  age += .5;  }  else if (currentMonth <= 6 && ((currentMonth + 6) >= birthMonth)) {  age += .5;  }  }  return age;  } | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TDD Development | | | | |
| Requirement: AO Rating Restriction | | | | |
| Test | Inputs | Expected Output | Actual Output | Pass/Fail |
| Same months | DOB: 2000-12-2  Today’s Date: 2018-12-1  Rating: AO | Age: 17  Customer cannot rent item. | Age: 17.5  Customer cannot rent item. | Fail |
| ‘’ | DOB: 2000-12-2  Today’s Date: 2018-12-3  Rating: AO | Age: 18  Customer can rent item. | Age: 18 | Pass |
| ‘’ | DOB: 2000-8-2  Today’s Date: 2018-8-2  Rating: AO | Age: 18  Customer can rent item. | Age: 18 | Pass |
| bool rentalValidation(int age, string rating) {  if (rating == "EC") {  return true;  }  else if (rating == "E") {  if (age >= 8) {  return true;  }  else return false;  }  else if (rating == "E10") {  if (age >= 10) {  return true;  }  else return false;  }  else if (rating == "T") {  if (age >= 13) {  return true;  }  else return false;  }  else if (rating == "M") {  if (age >= 17) {  return true;  }  else return false;  }  else if (rating == "AO") {  if (age >= 18) {  return true;  }  else return false;  }  } | | | | |

Assumptions:

The assumption made when writing this test plan was that the program would take a date of birth as a year, month, and day, and current date as a year, month, and day, then calculate the age based on the customers’ date of birth, display the age, then take a rating and make a comparison with the age to correctly determine whether the rental could go through or not. If a person is more than 6 months past their birthday, not considering days, .5 will be added to their age. For these test cases, inputs will be acquired by a CLI. For simplicity, today’s date will also be set by the tester. The program will not allow for birth years below 1900. The program has no lower age limit, meaning if a customer was born on today’s date or earlier it will accept the data.

//Checkout.cpp

#include <ctime>

#include <iostream>

#include <string>

using namespace std;

int main() {

int leapYears[35] = { 1904, 1908, 1912, 1916, 1920, 1924, 1928, 1932, 1936, 1940, 1944, 1948, 1952, 1956, 1960, 1964, 1968, 1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016, 2020, 2024, 2028, 2032, 2036, 2040 };

int birthMonth = -1, birthDay = -1, birthYear = -1, currentMonth, currentDay, currentYear;

bool isLeapYear = false;

double age = -1;

string rating = "";

bool canRent = false;

bool rentalValidation(double age, string rating);

double ageCalculator(int birthMonth, int birthDay, int birthYear, int currentMonth, int currentDay, int currentYear);

while (age < 0) {

birthYear = -1;

birthMonth = -1;

birthDay = -1;

while (birthYear < 1900 && age < 0) {

cout << "Enter customer birth year: ";

cin >> birthYear;

if (birthYear < 1900) {

cout << "Enter a valid birth year.\n";

}

}

while (birthMonth < 1 || birthMonth > 12 && age < 0) {

cout << "\nEnter customer birth month: ";

cin >> birthMonth;

if (birthMonth < 1 || birthMonth > 12) {

cout << "Enter a valid birth month.";

}

}

while (birthDay == -1) {

cout << "\nEnter customer birth day: ";

cin >> birthDay;

if (birthMonth == 2 && (birthDay < 1 || birthDay > 28)) {

if (birthDay == 29) {

for (int i = 0; i < 35; i++) {

if (birthYear == leapYears[i]) {

isLeapYear = true;

}

}

}

if (birthDay != 29 || (birthDay == 29 && !isLeapYear)) {

cout << "Enter a valid birth day.";

birthDay = -1;

}

}

if ((birthMonth == 4 || birthMonth == 6 || birthMonth == 9 || birthMonth == 11) && (birthDay < 1 || birthDay > 30)) {

cout << "Enter a valid birth day.";

birthDay = -1;

}

if ((birthMonth == 1 || birthMonth == 3 || birthMonth == 5 || birthMonth == 7 || birthMonth == 8 || birthMonth == 10 || birthMonth == 12) && (birthDay < 1 || birthDay > 31)) {

cout << "Enter a valid birth day.";

birthDay = -1;

}

}

cout << "\nEnter current year: ";

cin >> currentYear;

cout << "\nEnter current month: ";

cin >> currentMonth;

cout << "\nEnter current day: ";

cin >> currentDay;

age = ageCalculator(birthMonth, birthDay, birthYear, currentMonth, currentDay, currentYear);

cout << "\nAge: " << age;

if (age < 0 || birthYear < 1900) {

cout << "\nInvalid entry.\n";

age = -1;

}

}

while (rating != "EC" && rating != "E" && rating != "E10" && rating != "T" && rating != "M" && rating != "AO") {

cout << "\nEnter a rating: ";

cin.ignore();

getline(cin, rating);

if (rating != "EC" && rating != "E" && rating != "E10" && rating != "T" && rating != "M" && rating != "AO") {

cout << "\nEnter a valid rating.";

}

}

canRent = rentalValidation(age, rating);

if (canRent == false) {

cout << "\nCustomer cannot rent item.\n";

}

else if (canRent == true) {

cout << "\nCustomer can rent item.\n";

}

system("pause");

}

double ageCalculator(int birthMonth, int birthDay, int birthYear, int currentMonth, int currentDay, int currentYear) {

double age = currentYear - birthYear - 1;

if ((birthMonth < currentMonth) || (birthMonth == currentMonth && birthDay <= currentDay)) {

age++;

}

if (birthMonth <= 6) {

if (currentMonth < birthMonth) {

age += .5;

}

if (currentMonth == birthMonth && currentDay < birthDay) {

age += .5;

}

if (currentMonth >= (birthMonth + 6)) {

age += .5;

}

}

else if (birthMonth > 6) {

if (currentMonth == birthMonth && currentDay < birthDay) {

age += .5;

}

else if (currentMonth <= 6 && ((currentMonth + 6) >= birthMonth)) {

age += .5;

}

}

return age;

}

bool rentalValidation(double age, string rating) {

if (rating == "EC") {

return true;

}

else if (rating == "E") {

if (age >= 8.5) {

return true;

}

else return false;

}

else if (rating == "E10") {

if (age >= 10) {

return true;

}

else return false;

}

else if (rating == "T") {

if (age >= 13) {

return true;

}

else return false;

}

else if (rating == "M") {

if (age >= 17) {

return true;

}

else return false;

}

else if (rating == "AO") {

if (age >= 18) {

return true;

}

else return false;

}

}