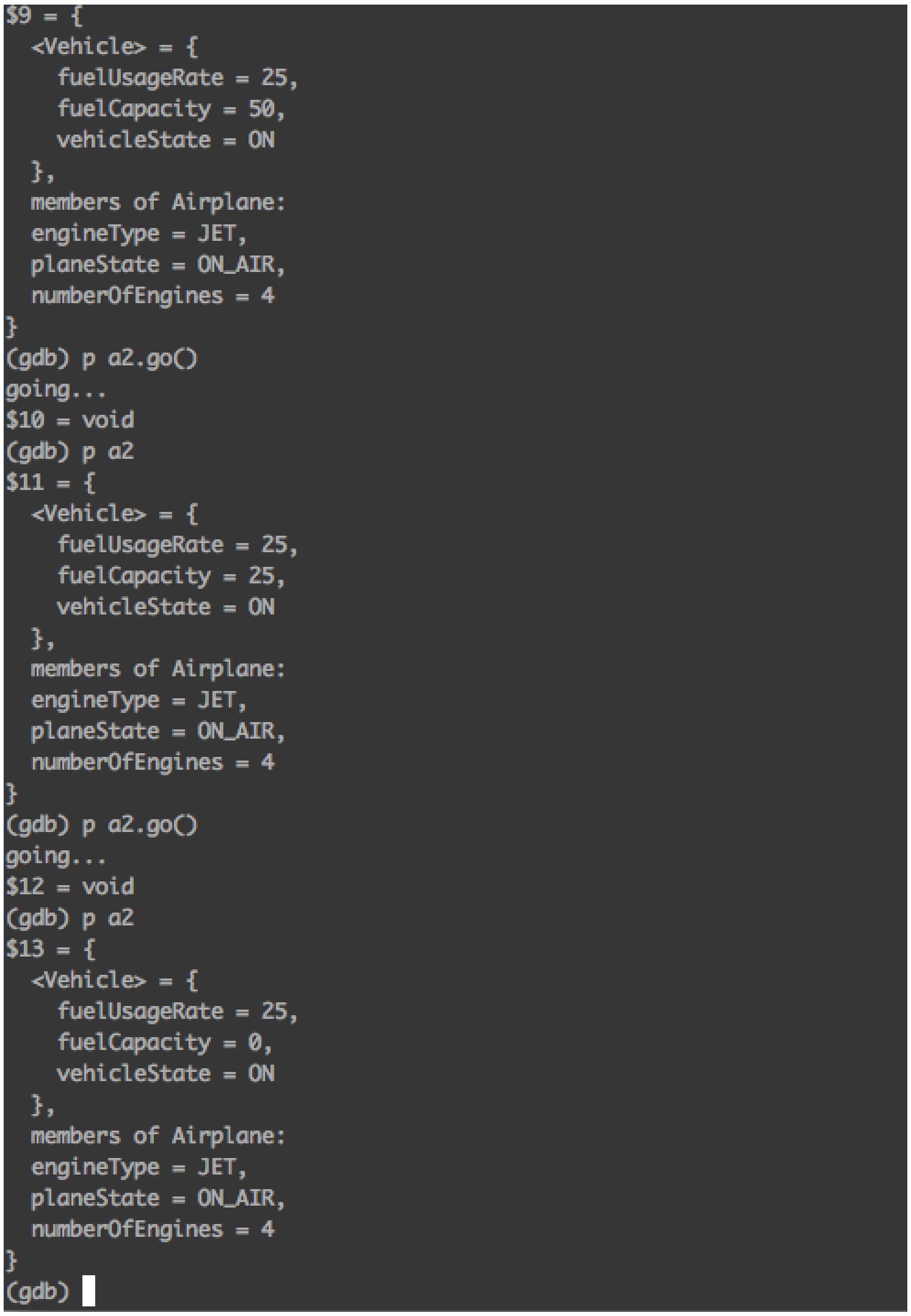
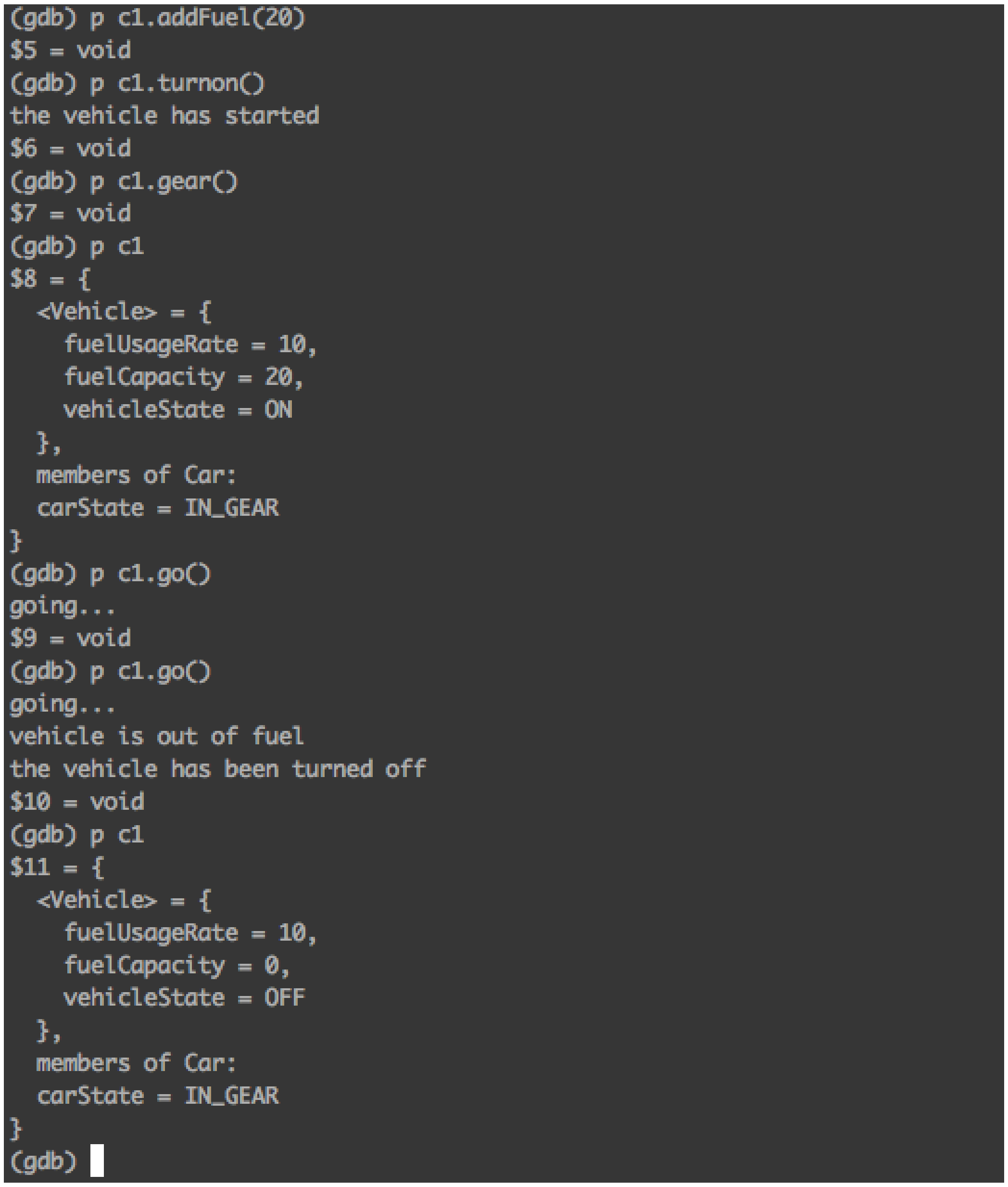
#Known Issues/Limitations

- ~~The plane does not crash when it runs out of fuel. Compiled program using the -g flag to debug. Under the debugger it appears that for some reason after consuming fuel the *if* condition that checks for if the fuel capacity is less than or equal to 0 return false. This only happens when the airplane runs out of fuel during the call to go method. When go method is called second time it prints message "out of fuel". (see below)~~



```
$9 = {
  <Vehicle> = {
    fuelUsageRate = 25,
    fuelCapacity = 50,
    vehicleState = ON
  },
  members of Airplane:
  engineType = JET,
  planeState = ON_AIR,
  numberOfEngines = 4
}
(gdb) p a2.go()
going...
$10 = void
(gdb) p a2
$11 = {
  <Vehicle> = {
    fuelUsageRate = 25,
    fuelCapacity = 25,
    vehicleState = ON
  },
  members of Airplane:
  engineType = JET,
  planeState = ON_AIR,
  numberOfEngines = 4
}
(gdb) p a2.go()
going...
$12 = void
(gdb) p a2
$13 = {
  <Vehicle> = {
    fuelUsageRate = 25,
    fuelCapacity = 0,
    vehicleState = ON
  },
  members of Airplane:
  engineType = JET,
  planeState = ON_AIR,
  numberOfEngines = 4
}
(gdb) 
```

- a possible limitation in the car class is that when *go* method is called until the fuel runs out, the car is turned of automatically. Even though the car is still *in gear* (see below).



```
(gdb) p c1.addFuel(20)
$5 = void
(gdb) p c1.turnon()
the vehicle has started
$6 = void
(gdb) p c1.gear()
$7 = void
(gdb) p c1
$8 = {
  <Vehicle> = {
    fuelUsageRate = 10,
    fuelCapacity = 20,
    vehicleState = ON
  },
  members of Car:
  carState = IN_GEAR
}
(gdb) p c1.go()
going...
$9 = void
(gdb) p c1.go()
going...
vehicle is out of fuel
the vehicle has been turned off
$10 = void
(gdb) p c1
$11 = {
  <Vehicle> = {
    fuelUsageRate = 10,
    fuelCapacity = 0,
    vehicleState = OFF
  },
  members of Car:
  carState = IN_GEAR
}
(gdb)
```

When viewed from real-world perspective this makes sense. But the specification says "*unless the car is in "park" then it cannot be turned off."* This can be easily fixed by adding more functionality to the overridden go method. Because this behavior make sense, I'm leaving this as-is.

- When creating objects of Airplane class, you pass in engine type as a string. Right now, the checks for engine type "jet" and "prop" and assign proper engine type to *engineType* data member. If some other string other than "jet" or "prop" is passed, I just assign PROP to *engineType* by default. I was going to throw an exception here, but at this time I don't really know how to do this in C++ yet. I plan on reading the chapter on exceptions soon.

#Algorithm

```
class Car < Vehicle

default_constructor Car
  initialize Vehicle
  set CAR_STATE to IN_PARK
end default_constructor

method turnoff
  if CAR_STATE is IN_GEAR
    print “park the car before turning off”

  if CAR_STATE is IN_PARK
    call_parent_method turnoff
end turnoff

method go
  if CAR_STATE is IN_GEAR
    call_parent_method go

  if CAR_STATE is IN_PARK
    print “the car is in park, you cannot go”
end go

method park
  set CAR_STATE to IN_PARK
end park

method gear
  set CAR_STATE to IN_GEAR
end gear

class Airplane < Vehicle

default_constructor Airplane
  initialize Vehicle
  set ENGINES to 1
  set ENGINE_TYPE to PROP
  set PLANE_STATE to ON_GROUND
end default_constructor

constructor Airplane(FUEL, FUEL_USAGE_RATE, INPUT_ENGINE_TYPE,
INPUT_ENGINE_NUMBER)
  initialize Vehicle with FUEL and FUEL_USAGE_RATE
  if “jet” equals to INPUT_ENGINE_TYPE
    set ENGINE_TYPE to JET
  else if “prop” equals to INPUT_ENGINE_TYPE
    set ENGINE_TYPE to PROP
  else
    //throw_exception
    set ENGINE_TYPE to PROP

  set PLANE_STATE to ON_GROUND
  set ENGINE_NUMBER to INPUT_ENGINE_NUMBER
end constructor



method takeoff:
  if VEHICLE_STATE is ON:
    A = fuel_usage_rate
    B = fuel_capacity
    C = fuel_usage_on_takeoff

    C = A*10

     if (B-C) <= 0:
       print “Not enough fuel to takeoff”
     else
        set B to (B-C)   // decrease fuel
        set PLANE_STATE to ON_AIR
        print “we are in the air”

  if VEHICLE_STATE is OFF:
        print “the airplane isn’t running so you cant takeoff”

end takeoff

method land:
  if PLANE_STATE is ON_AIR:
    set PLANE_STATE to ON_GROUND
    print “the plane has landed”

  if PLANE_STATE is ON_GROUND
        print “plane is already on the ground”
end land

method go:
  if VEHICLE_STATE is ON and PLANE_STATE is ON_AIR:
    A = fuel_usage_rate
    B = fuel_capacity

    if B > 0:
      set B to B - A  // decrease fuel
      print “going...”

    if B < 0:
      print “vehicle is out of fuel”
      set B to 0 // Because this could be negative
      call_method turnoff()

  if PLANE_STATE is ON_GROUND:
     print “not in the air”

  if VEHICLE_STATE is OFF:
     print “the vehicle has not been tuned on”
end go

method turnoff:
  call_parent_method turnoff

  if PLANE_STATE is ON_AIR:
    print “the plane has crashed”
    set PLANE_STATE to ON_GROUND;
end turnoff

class Vehicle
method turnon:
  if the vehicle is OFF and amount of FUEL is greater than 0:
     turn vehicle on
     print “the vehicle has started”

  if the vehicle is ON:
     print “the vehicle is already on”

  if no FUEL:
     print “there is no fuel in the vehicle”
end turnon

method turnoff:
  if vehicle is ON:
     turn vehicle OFF
     print “the vehicle has been turned off”
  if vehicle is OFF:
     print “the vehicle is not running”
end turnoff

method go:
  if vehicle is ON and there is FUEL:
     decrease FUEL_CAPACITY by FUEL_USAGE_RATE
     print “going...”

  if no FUEL:
     print “vehicle is out of fuel”
     reset FUEL to 0 //This is because FUEL variable could be negative
     turn vehicle OFF
end go

method addFuel FUEL_TO_BE_ADDED:
  if FUEL_TO_BE_ADDED is less than or equal to 0:
     print “cannot take away fuel”

  if FUEL_TO_BE_ADDED is greater than 0:
     add FUEL_TO_BE_ADDED to FUEL_CAPACITY
end addFuel

method getFuelUsageRate
  return FUEL_USAGE_RATE
end getFuelusageRate

method getFuelCapacity
  return FUEL_CAPACITY
end getFuelCapacity

method setFuelCapacity (int INPUT_CAPACITY)
  set FUEL_CAPACITY to INPUT_CAPACITY
end setFuelCapacity

method setVehicleState (State INPUT_STATE)
  set VEHICLE_STATE to INPUT_STATE
end setVehicleState
```