```
/**
 * @brief pop operation, popping the latest promoted model off the stack
 *        Both time and auxiliary space complexity need to be O(1)
 * @param
 * @return PromotedModel
 */
PromotedModel PromotedCarModelStack::pop() {

  //Makes sure the carNameAndPrice vector isn't empty
  if (carNameAndPrice.empty()){
    throw std::logic_error("Promoted car model stack is empty");
  }

  //stores last item of carNameAndPrice vector for use
  topOfCarStack = carNameAndPrice.back();

  //Checks to see if the last PromotedModel object in carNameAndPrice matches both the max & min prices
  //Finds and saves the next min and max prices
  if ((topOfCarStack.getPromotedPrice() == maxModel.getPromotedPrice()) && (topOfCarStack.getPromotedPrice() == minModel.getPromotedPrice())){
    poppedItem = carNameAndPrice.back();
    carNameAndPrice.pop_back();
    highestAndLowestPrice.pop_back();
    highestAndLowestPrice.pop_back();

    minModel = highestAndLowestPrice.back();
    maxModel = highestAndLowestPrice.at(highestAndLowestPrice.size() - 2);
  } else {

    //Checks to see if the last object just matches the min price & pops one item from carNameAndPrice while two for highestAndLowestPrice
    //Finds and saves the next min price
    if (topOfCarStack.getPromotedPrice() == minModel.getPromotedPrice()){
      poppedItem = carNameAndPrice.back();
      carNameAndPrice.pop_back();
      highestAndLowestPrice.pop_back();
      highestAndLowestPrice.pop_back();

      minModel = highestAndLowestPrice.back();
    }

    //Checks to see if the last object just matches the max price & pops one item from carNameAndPrice while two for highestAndLowestPrice
    //Finds and saves the next max price
    if (topOfCarStack.getPromotedPrice() == maxModel.getPromotedPrice()) {
      poppedItem = carNameAndPrice.back();
      carNameAndPrice.pop_back();
      highestAndLowestPrice.pop_back();
      highestAndLowestPrice.pop_back();

      maxModel = highestAndLowestPrice.at(highestAndLowestPrice.size() - 2);
    }

  }

  //Checks to see if the last object matches neither the max or the min prices & pops one item from carNameAndPrice while two for highestAndLowestPrice
  if (!(topOfCarStack.getPromotedPrice() == maxModel.getPromotedPrice()) && !(topOfCarStack.getPromotedPrice() == minModel.getPromotedPrice())){
    poppedItem = carNameAndPrice.back();
    carNameAndPrice.pop_back();
    highestAndLowestPrice.pop_back();
    highestAndLowestPrice.pop_back();
  }


  return poppedItem;
}
```