```
void PromotedCarModelStack::push(string model, int price) {

 //Checks to see if carNameAndPrice vector is empty & creates car, maxMode, & minModel objects to be inserted into their respective vectors
 if (carNameAndPrice.empty()){
   car = PromotedModel(model, price);
   maxModel = PromotedModel(model, price);
   minModel = PromotedModel(model, price);

   carNameAndPrice.push_back(car);
   highestAndLowestPrice.push_back(maxModel);
   highestAndLowestPrice.push_back(minModel);
 }

 /*
 * If passed in price is greater than maxModel's price, create a new PromotedModel car &
 * replace maxModel with new price
 *
 * Input car into carNameAndPrice along with the new price for highestAndLowestPrice
 */
 if (price > maxModel.getPromotedPrice()){ +2
   car = PromotedModel(model, price);
   carNameAndPrice.push_back(car);
   maxModel = PromotedModel(model, price);
   highestAndLowestPrice.push_back(maxModel);
   highestAndLowestPrice.push_back(minModel);
 }

 /*
 * If passed in price is less than minModel's price, create a new PromotedModel car &
 * replace minModel with new price
 *
 * Input car into carNameAndPrice along with the new price for highestAndLowestPrice
 */
 if (price < minModel.getPromotedPrice()){ +2
   car = PromotedModel(model, price);
   carNameAndPrice.push_back(car);
   minModel = PromotedModel(model, price);
   highestAndLowestPrice.push_back(maxModel);
   highestAndLowestPrice.push_back(minModel);
 }

 /*
 * If the price is not greater or less than the min/maxModel then create PromotedModel car,
 * input that into carNameAndPrice along with the same min/maxModels into highestAndLowestPrice
 */
 if(!(price >= maxModel.getPromotedPrice()) && !(price <= minModel.getPromotedPrice())){ +7
   car = PromotedModel(model, price);
   carNameAndPrice.push_back(car);
   highestAndLowestPrice.push_back(maxModel);
   highestAndLowestPrice.push_back(minModel);
 }


}
```

1+1+1+1+1+1 = 6

1+1+1+1+1 = 5

1+1+1+1+1 = 5

1+1+1+1 = 4