Task- provide detailed gas cost report for the following code by using storage, memory and calldata for the different structs and variables used. Use separate functions to get title, author, bookId and price for this task.

```
//SPDX-License-Identifier:MIT
pragma solidity 0.8.6;
contract structure{
struct Book{
string title;
string author;
uint bookID;
uint price;
}
// define a struct- name of the struct variable to represent the struct
Book book;
function setBook() public {
book= Book("Blokchain for beginners","Ineuron",4,1000);
}
function getBookId() public view returns(uint){
return book.bookID;
}
```

ANS:

We must examine the gas expenses related to the use of storage, memory, and calldata for the various structs and variables used in the contract in order to produce a thorough gas cost report for the provided code. Data manipulation, function calls, and storage activities are only a few of the variables that affect petrol prices.
Let's dissect the code and examine how much petrol each operation costs:

1. Definition of a Contract:
• Gas cost: Negligible as it's only defining the contract.

2. Definition Structure:
Gas cost: Since it's merely defining a struct, it's negligible.

3. Statement of State Variable:
• Gas cost: Declaring a state variable book of type Book in storage has a negligible gas cost.

setBook() Function:

Gas cost: Data is initialised in the book struct by the setBook() function. This function will use more gas since it writes to storage, which increases storage writes. The quantity of data being written and the number of state variables being written to will be the main determinants of the petrol cost. Writing to book.title, book.author, book.bookID, and book.price is the storage method.

Calldata: This function does not use any calldata operations.
Memory: This function does not require any major memory operations.
obtainBookId() Use:

Gas cost: The getBookId() function uses storage to read data, which has a cost associated with it but is typically less than writing data.
Reading a book as storage.bookID.
Calldata: This function does not use any calldata operations.
Memory: This function does not require any major memory operations.

The gas costs associated with different EVM operations, such as SSTORE (storage write), SLOAD (storage read), CALLDATALOAD (calldata read), etc., as listed in the Ethereum Yellow Paper, must be consulted in order to provide a more thorough analysis of gas prices.

Use tools like Remix IDE, Truffle, or Hardhat, which frequently provide gas predictions for various activities within your contract, for a more accurate report on gas costs. To determine how much gas will be used for deploying and carrying out the tasks in your contract, you can perform simulations. Furthermore, based on existing network circumstances, tools such as Gas Station Network (GSN) can offer more precise gas cost estimates.