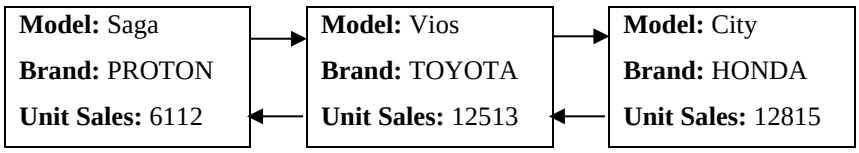
QUESTION 1 – LINKED LIST [50 Marks]

Given a C++ program file (`Test2-1cs.cpp`) to create and manage a list of car sales information. Inside the program, the car sale information is defined by the `CarSale` class, while the `DoublyLinkedList` class is used to store and manage the `CarSale` class objects as items in a doubly linked list. The `initializeList(CarSale *sales[], int size)` method of the `DoublyLinkedList` class has been provided to initially connect the car sales, as shown below:



The `CarSale` class has three member variables (`model`, `brand`, and `unitSale`) and one method (`printSaleInfo()`) to represent the car sale information. Within the `DoublyLinkedList` class, the `head` and `tail` member variables are pointers that respectively point to the first and last `CarSale` objects, while the `next` and `prev` member variables are pointers used to connect them as doubly linked list items.

Pre-requisite Tasks:

1. Open the program and complete the following section with your profile info:

```
1  // Test 2 - SECJ2013 - 24251 (Test2-1cs.cpp)
2  // Name: ???
3  // Matric No.: ???
4  // Section: ???
```

2. Compile and run the program. The program should produce the following output:

```
Add Saga to the list...
Add Vios to the list...
Add City to the list...

Task 1: Display Forward:

Task 2: Display Backward:

Task 3: Add 'Hilux' Between 'Saga' and 'Vios':

Display Forward:

Display Backward:

Task 4: Delete last node:

Display Forward:
```

```
Display Backward:

Task 5: Delete and replace first node with 'Bezza':

Display Forward:

Display Backward:

Task 6: Display forward, delete and total up unit sales:
```

Main Tasks:

Append the appropriate lines of code at the space provided in `Test2-1cs.cpp` to accomplish the following main tasks. Note that, each task is continuous from one to the next. The expected state of the list after the execution of each task is given in each question/task below (Expected Output). You must not change any line of code related to the class definition or the provided lines of code in the `main` function unless instructed by the question.

Task1

Complete the `displayForward()` function of `DoublyLinkedList` class to displays all the car sales information in the order they were firstly added into the list. You can call the `printSaleInfo()` function/method of `CarSale` class to print the car sale information.

(5 marks)

Expected Output of Task 1:

```
Task 1: Display Forward:
Saga (PROTON) - 6112 units
Vios (TOYOTA) - 12513 units
City (HONDA) - 12815 units
```

Task 2

Complete the `displayBackward()` function that displays all the car sales information in the reverse order.

(5 marks)

Expected Output of Task 2:

```
Task 2: Display Backward:
City (HONDA) - 12815 units
Vios (TOYOTA) - 12513 units
Saga (PROTON) - 6112 units
```

Task 3

Complete the `addBetween()` function to add the `Hilux` car sale information between `Saga` and `Vios`. Below is the information to be used for the `Hilux` car sale.

Model: Hilux
Brand: TOYOTA
Unit Sales: 11540

(12 marks)

Expected Output of Task 3:

```
Task 3: Add 'Hilux' between 'Saga' and 'Vios':

Display Forward:
Saga (PROTON) - 6112 units
Hilux (TOYOTA) - 11540 units
Vios (TOYOTA) - 12513 units
City (HONDA) - 12815 units

Display Backward:
City (HONDA) - 12815 units
Vios (TOYOTA) - 12513 units
Hilux (TOYOTA) - 11540 units
Saga (PROTON) - 6112 units
```

Task 4

Complete the `deleteLast()` function to delete the last node from the list.

(7 marks)

Expected Output of Task 4:

```
Task 4: Delete last node:
Destroy CarSale: City

Display Forward:
Saga (PROTON) - 6112 units
Hilux (TOYOTA) - 11540 units
Vios (TOYOTA) - 12513 units

Display Backward:
Vios (TOYOTA) - 12513 units
Hilux (TOYOTA) - 11540 units
Saga (PROTON) - 6112 units
```

Task 5

Complete the `replaceFirst()` function to delete and replace the first node (`Saga`) with `Bezza`. Below is the information to be used for the `Bezza` car sale.

Model: Bezza
Brand: PERODUA
Unit Sales: 47100

(8 marks)

Expected Output of Task 5:

```
Task 5: Delete and replace first node with 'Bezza':
Destroy CarSale: Saga

Display Forward:
Bezza (PERODUA) - 47100 units
Hilux (TOYOTA) - 11540 units
Vios (TOYOTA) - 12513 units

Display Backward:
Vios (TOYOTA) - 12513 units
Hilux (TOYOTA) - 11540 units
Bezza (PERODUA) - 47100 units
```

Task 6

Complete the `displayAndTotal()` function to:

i) Displays and deletes all the `CarSale` objects from the linked list in forward order.

ii) Prints the total unit sales of all car model.

(13 marks)

Expected Output of Task 6:

```
Task 6: Display forward, delete and total up unit sales:
Bezza (PERODUA) - 47100 units
Destroy CarSale: Bezza
Hilux (TOYOTA) - 11540 units
Destroy CarSale: Hilux
Vios (TOYOTA) - 12513 units
Destroy CarSale: Vios

TOTAL SALE = 71153 units
```

QUESTION 2 – STACK [50 Marks]

Given a C++ program (`Test2-2cl.cpp`) to implements a stack using an array. The program contains 2 classes (`Cylinder` and `Stack`). The `Cylinder` class stores information about a cylinder, including three attributes: `ID`, `radius`, and `height`. It also has three methods: `getID()`, `volume()`, and `printCylinderInfo()`.

In the `Stack` class, the `Cylinder data[STACK_SIZE]` member variable is defined to stores `Cylinder` objects in an array. This array acts as the internal storage for the stack, enabling push and pop operations on `Cylinder` objects.

Study how the `main` function is currently used to initialize and managing `Cylinder` objects using the `Stack` class. Complete the lines of code in the `Test2-2cl.cpp` program by following the upcoming tasks. You must not change any line of code related to the class definition or the provided lines of code in the `main` function unless instructed by the question.

Pre-requisite Tasks:

1. Open the program and complete the following section with your profile info:

```
1 // Test 2 - SECJ2013 - 24251 (Test2-2cl.cpp)
2 // Name: ???
3 // Matric No.: ???
4 // Section: ???
```

2. Compile and run the program. The program should produce the following output:

```
Try to push all cylinders onto the stack:

Cylinder at top => CL0000, 0

Current stack content:


Pop stack until Cylinder 'CL4399' becomes the top and
then push Cylinder 'CL4350' onto the stack...
Current stack content:

Pop all cylinders from stack:

Try to pop Cylinder even stack is empty:
```

Main Tasks:

Task 1:

Complete the implementation of `createStack()` method of the `Stack` class to correctly initialize its member data for stack implementation using array.

(2 marks)

Task 2:

Complete the implementation of `push(Cylinder cl)` method of the `Stack` class to push `Cylinder` object onto the stack. Below is an example of the outputs of the method for successful and failed push operations.

Example of output for successful push operation:

```
Push Cylinder CL3215 onto stack.
```

Example of output for failed push operation:

```
Can't push Cylinder CL4350, stack is full!
```

(12 marks)

Task 3:

Complete the implementation of `pop()` method of the `Stack` class to pop `Cylinder` object from the stack. Below is an example of the outputs of the method for successful and failed pop operations.

Example of output for successful pop operation:

```
Pop Cylinder CL8480 from stack
```

Example of output for failed pop operation:

```
Can't pop Cylinder from stack, stack is empty!
```

(12 marks)

Task 4:

Modify the implementation of `stackTop()`, `isEmpty()`, and `isFull()` methods of the `Stack` class to correctly return the `Cylinder` object at the top of the stack and determine either the stack is full or empty.

(6 marks)

As a guide, below is the new output of the program after you have successfully completed Tasks 1-4:

```
Try to push all cylinders onto the stack:
Push Cylinder CL3215 onto stack.
Push Cylinder CL9253 onto stack.
Push Cylinder CL4399 onto stack.
Push Cylinder CL3200 onto stack.
Push Cylinder CL8480 onto stack.
Can't push Cylinder CL4350, stack is full!
```

```
Cylinder at top => CL8480, 529875

Current stack content:
Top -> CL8480, 529875
       CL3200, 20403
       CL4399, 8478
       CL9253, 29673
       CL3215, 226080


Pop stack until Cylinder 'CL4399' becomes the top and
then push Cylinder 'CL4350' onto the stack...
Current stack content:
Top -> CL8480, 529875
       CL3200, 20403
       CL4399, 8478
       CL9253, 29673
       CL3215, 226080

Pop all cylinders from stack:
Pop Cylinder CL8480 from stack
Pop Cylinder CL3200 from stack
Pop Cylinder CL4399 from stack
Pop Cylinder CL9253 from stack
Pop Cylinder CL3215 from stack

Try to pop Cylinder even stack is empty:
Can't pop Cylinder from stack, stack is empty!
```

Task 5:

Inside the `main` function, locate the section with the following comments/instructions:

```
// pop cylinder objects using 'while' loop until
// Cylinder 'CL4399' becomes the top
// ???
// ???
// ???

// push Cylinder 'CL4350' onto stack
// ???
```

Complete the code in the above section, guided by the given comments/instructions.

(18 marks)

Below is the final output of the program after successfully completing Tasks 1-5:

```
Try to push all cylinders onto the stack:
Push Cylinder CL3215 onto stack.
Push Cylinder CL9253 onto stack.
Push Cylinder CL4399 onto stack.
Push Cylinder CL3200 onto stack.
Push Cylinder CL8480 onto stack.
Can't push Cylinder CL4350, stack is full!

Cylinder at top => CL8480, 529875

Current stack content:
Top -> CL8480, 529875
       CL3200, 20403
       CL4399, 8478
       CL9253, 29673
       CL3215, 226080


Pop stack until Cylinder 'CL4399' becomes the top and
then push Cylinder 'CL4350' onto the stack...
Pop Cylinder CL8480 from stack
Pop Cylinder CL3200 from stack
Push Cylinder CL4350 onto stack.
Current stack content:
Top -> CL4350, 15825
       CL4399, 8478
       CL9253, 29673
       CL3215, 226080

Pop all cylinders from stack:
Pop Cylinder CL4350 from stack
Pop Cylinder CL4399 from stack
Pop Cylinder CL9253 from stack
Pop Cylinder CL3215 from stack

Try to pop Cylinder even stack is empty:
Can't pop Cylinder from stack, stack is empty!
```

End of Question