Deque

What is Deque?

- Deque is like a dynamic array with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.
- They support insertion and deletion from both the ends in an amortized constant time - O(1).
- Inserting and erasing in the middle is linear in time - O(n).

What is expected as a solution?

- The C++ standard specifies that a legal implementation of deque must satisfy the following requirements:

Deque

```
Deque()                - initialize a blank deque.
Deque()                - initialize a deque of length n with all values as x.
push_back(x)           - append data x at the end.
pop_back()             - erase data at the end.
push_front(x)          - append data x at the beginning.
pop_front()            - erase data at the beginning.
front()                - returns the first element(value) in the deque.
back()                 - returns the last element(value) in the deque.
empty()                - returns true if deque is empty else returns false.
size()                 - returns the current size of deque.
resize(s,d,direction) - Changes the size dynamically. 's' is the new size of the
deque and 'direction' is either 1(forward) or -1(reverse). If the new size is
greater than the current size of the deque, then increase the size at the end (if
direction is set to 1) or increase the size at the beginning of the deque (if the
direction value is set to -1) and then fill the empty space with the default value
d. If the new size of the deque is less than the current size then remove elements
at the end (if direction = 1) or remove elements at the beginning (if direction = -
1).
clear()                - remove all elements of deque.
D[n]                   - returns the nth element of the deque.
```

DequeMock (for testing the Deque)

```
DequeMock()             - Constructor
print()                 - print all the elements in the deque.
processDeque()          - perform operations related to deque
displayMenu()           - display menu(s) for user operations
...
...
<you can add any number of functions for testing purpose based on your test
requirements>.
```

Testing the Deque

Menu driven approach for input:

```
What type of deque you want to create:
1. Integer
2. String
3. User defined (Student)
Your Choice:
```

Once you get the data type from the user then have the following menu:

```
-------------------------------------------------------------------------------
|                                    Deque                                    |
-------------------------------------------------------------------------------
| 1.  Create an empty deque                                                   |
| 2.  Create a deque with 'n' elements with default value of 'x'              |
| 3.  Add an element at end                                                   |
| 4.  Add an element at the beginning                                         |
| 5.  Remove an element from end                                              |
| 6.  Remove an element from beginning                                        |
| 7.  Get the first element                                                   |
| 8.  Get the last element                                                    |
| 9.  Get the size of deque                                                   |
| 10. Resize the deque to new size 's' with default value 'd' and ‘direction’.  |
| 11. Clear all the elements in the deque                                     |
| 12. Print all the elements in the deque                                     |
| 12. Enter -1 to exit.                                                       |
-------------------------------------------------------------------------------
Your Choice:
```

Based on the choice take required input from the user.

The above Deque menu should be repeatedly displayed until -1 is entered.

Creation of Mock class for testing:

- For testing your Deque class you need to create a DequeMock class for testing.
- class DequeMock : public Deque
- Add some additional functions like – “displayMenu(), procesDeque(), etc” … to work on the Deque and get results.
- So, the driver code to test the Deque should actually create a DequeMock object and run the tests.
- Reach out to me for any help in understanding.

Evaluation Parameters:

- Accuracy of operations, memory management and performance.

Important Guidelines for implementation:

- Use only C++ as programming language.
- DO NOT use any of the C++ STL libraries like vectors/maps/queue etc. You need to implement your own data structures and handle the memory management yourself.
- The deque implemented should be generic which should support any data type, like int/float/string or any user defined data types like objects instantiated from any structure or class (Hint: use templates).
- Follow PHYP Coding guidelines for naming conventions and others coding details.
- Write appropriate test cases to test your code. Try all the data types int/float/double/string. Also create some structure - for.eg: create a Student class/structure with member variables like {name, roll-no, section, gender, etc} and then try to create multiple student objects and perform all the operations of deque on it.
- PLAGIARISM is strictly prohibited and online solutions from websites like stack-exchange etc, will not be tolerated. I can figure out if the solution is copied from somewhere. So, avoid such short-cuts and work on the problem on your own.
- Avoid using online C++ compilers. Install "command-line-tools" in your macbook (which will install g++ compiler) and use it.
- Ask for help if needed.