temp = 61 − 7 = 54   −6   7   −6   7+10+1   18

temp
20 − 49

CC-214-L: Object Oriented Programming Lab
BSSE Fall 2020
Lab 05

−6 + 67
61

Issue Date: 05-Nov-2021
Marks: 16

2
18
18
18
54

Objective:

tho
119 − 59

- Object's initialization and a bit of logic as always to keep your brains working.
- Distributing code in header and cpp files.

0 + 120
120 − 59
61

−10 −9 −8 −7 −6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6 7

Challenge-A: *Retail Item*

Write a class named `RetailItem` that holds data about an item in a retail store. The class should have the following member variables:

3

- `description`: A `char array` that holds a brief description of the item.
- `unitsInHand`: An `int` that holds the number of units currently in inventory.
- `price`: A `double` that holds the item s retail price.

Write a constructor that accepts arguments for each data member, appropriate mutator/setter functions that store values in these member variables, and accessor/getter functions that return the values in these member variables. Once you have written the `class`, write a separate program that creates three `RetailItem` objects and stores the following data in them.

	Description	Units in Hand	Price
Item # 1	Jacket	12	59.95
Item # 2	Designer Jeans	40	34
Item # 3	Shirt	20	24.25

Challenge-B: *Bounded Integral Type*

In your programming career, you may not always get ready to use data types which works as per your program logic/requirements. Sometimes, we have to design/mold the language to define types so that we may get a clean/easy support for the problem/solution under consideration.
For example, if we were writing a program to implement a clock or a calendar, then we would need numbers to represent minutes or days. But these numbers aren't C/C++ integers, which range from (-2^{31} to $2^{31}-1$); they range from 0 to 59 and 1 to 31 (or less!), respectively. C/C++ primitive types are useful building blocks, but at the level of atoms, not molecules.

For this task, you need to develop an ADT named as `'BoundedInteger'` which store values within a given range.

The most important feature of our *`BoundedInteger`* class is to wrap-around when we try to increment or decrement some given value in it.
For example, suppose that we have a *`minutes`* bounded integer (0~59) whose value is 52. Adding 10 to this object gives it a value of 2. See the sample run for further understanding.
The detail below gives explanation of how the `'BoundedInteger'` ADT members will act to achieve the desired objective.

Data Members:
The first two members i.e., *`lowerBound`* and *`upperBound`* represents the range of data that can be stored in a bounded integer object.
The *`value`* represents the integral-value stored in bounded integer which should be in the range *`(lowerBound <= value <= upperBound)`*

- `lowerBound;`
- `upperBound;`
- `value;`

3
0 −1

Member Functions:

- `bool isValidBound(int lb, int ub);`
  `A utility function which helps you find whether the bound is valid or not. i.e., it should be: (lb <= ub).`
- `BoundedInteger();`
  `Constructor, which initializes the lower Bound with INT_MIN and upper bound with INT_MAX, and set the value to lower bound. INT_MIN and INT_MAX are predefined constants defined in C++, where INT_MIN is the minimum value in int and INT_MAX is the maximum value in int.`

$\boxed{7}$

- `BoundedInteger(int lb, int ub);`
  It initializes the lower bound and upper bound as received in parameters. It should validate the bounds and also initializes the value with lower bound. If the bound is not valid then it initializes as per default constructor.
- `BoundedInteger(int lb, int ub, int val);`
  Same as the previous constructor except it also receive the val to set with data member value.
- `void setValue(int val);`
  setter for value.
- `int getValue();`
  getter for value.
- `int getLowerBound();`
  getter for lowerBound
- `int getUpperBound();`
  getter for upperBound
- `void increment(int inc=1);`
  increments/adds the received inc in the value keeping in view the bounds and wraps around accordingly. Assumption: inc value will always be >= 0.
- `void decrement(int dec=1);`
  decrements/subtracts the received dec from the value keeping in view the bounds and wraps around accordingly. Assumption: dec value will always be >= 0.
- `bool isValidValue(int val);`
  validates, whether the received val is valid according to bounds or not.

-10 − 13+   -10   7   -6   ~~-6 − 61~~ 1 +
16) 61 (3   48 $\boxed{13}$ -16   61
-6 + 67   -10 − 7 + 1

```
//BoundedInteger.h
class BoundedInteger
{
      int lowerBound;
      int upperBound;
      int value;
      bool isValidBound(int lb, int ub);

public:

      BoundedInteger();
      BoundedInteger(int lb, int ub);
      BoundedInteger(int lb, int ub, int val);
      void setValue(int val);
      int getValue();
      int getLowerBound();
      int getUpperBound();
      void increment(int inc=1);
      void decrement(int dec=1);
      bool isValidValue(int val);
};
```

```
//BoundedInteger.cpp
/*
Definitions for BoundedInteger.h
*/

//main.cpp
int main()
{
      BoundedInteger bi1(5,12,7);
      bi1.decrement(4);
      cout<<bi1.getValue()<<'\n';//prints 11
      bi1.increment(10);
      cout<<bi1.getValue()<<'\n';//prints 5

      BoundedInteger bi2(-20, 10, 100);
      bi2.increment(3);
      cout<<bi2.getValue()<<'\n';//prints -17

      return 0;
}
```

5 − 3   2 $\boxed{2 \% 10}$ 7 − 4   5 6 7 8 9 10 11 12   7 − 4   3   21   21 % 9   11 + 10   21   21 % 12   15   9   5 − 10   21 − 12   9 % 9   49 50 51 52 53 … 59   21 − 12   9 % 8

Note: While developing logic for increment/decrement function, you may go for applying looping mechanism but that will not give you full credit rather very less marks. So, think smart.

2 − 59   1 − 59   59 + 6   65 % 2 = 1   65 % 1 = 0   1 + 2 − 1 = $\boxed{2}$ 0 + 1 − 1 = 0

0 − 59   2   -20   10   0 − 65 % 1   -17   % -20   -20 + 3   3 $\boxed{-17}$ 1 − 59   -17   -2   -20 + 3   20   20 % 1   65 % 1   0 + 1 − 1 = 0

I HATED EVERY MINUTE OF TRAINING, BUT I SAID, 'DON'T QUIT. SUFFER NOW AND LIVE THE REST OF YOUR LIFE AS A CHAMPION.

[--- MUHAMMAD ALI ---]
[17-JAN-1942 – 3-JUNE-2016]

0 1 2 3 4 5 6 7 8 9