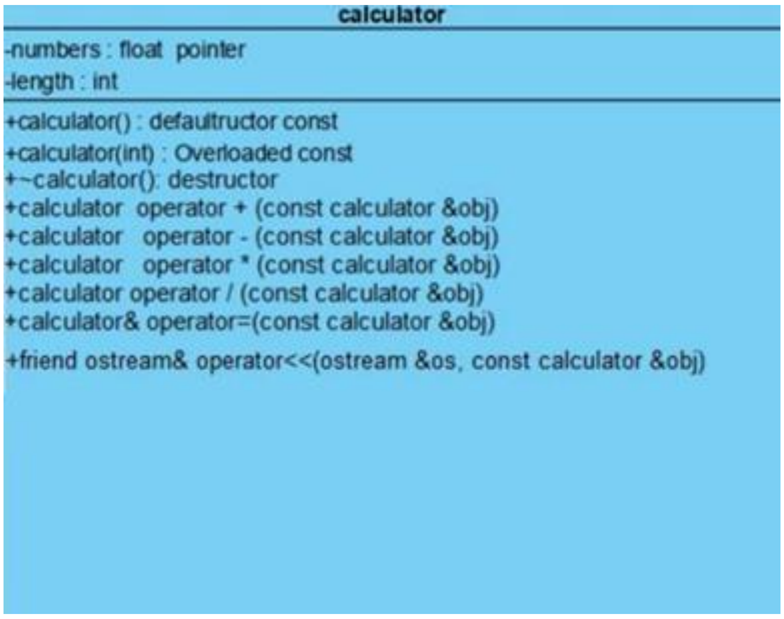
CENG 241 LAB 9

1-) Write a c++ program to make a calculator to operate on arrays. Calculator needs to have operations: +, -, /, *. Calculator class is like following:



calculator

-numbers : float pointer
-length : int

+calculator() : defaultructor const
+calculator(int) : Overloaded const
+~calculator(): destructor
+calculator operator + (const calculator &obj)
+calculator operator - (const calculator &obj)
+calculator operator * (const calculator &obj)
+calculator operator / (const calculator &obj)
+calculator& operator=(const calculator &obj)
+friend ostream& operator<<(ostream &os, const calculator &obj)

Class variables:

- numbers: You will have a dynamic array for the numbers entered.

- Length: This variable stores the size of array.

Class Functions:

- Calculator(): Default Constructor for your class.

- Calculator(int): Overloaded constructor that takes one parameter for the size of array.

- ~calculator(): In this destructor function, you will delete your dynamic array.

Overloaded Operators:

- calculator operator + (const calculator &obj): This is an overloaded operator to sum two dynamic arrays of different objects.

- calculator operator – (const calculator &obj): This is an overloaded operator to subtract two dynamic arrays of different objects.
- calculator operator * (const calculator &obj): This is an overloaded operator to multiply two dynamic arrays of different objects by each other.
- calculator operator / (const calculator &obj): This is an overloaded operator to divide two dynamic arrays of different objects by each other.
- friend ostream& operator<< (ostream &os, const calculator &obj): This is a friend function to print arrays. Thanks to this function you can print your arrays with this format: "cout<< object;"
- = operator : works just like copy constructor, copying one object to the other.

Your program will ask user how many numbers of sets he wants to input then you determine the dynamic array in your class. Then, you will ask for second number of sets to enter. After all you will let user to choose the arithmetic operation to calculate. Then you will print the result with your friend function.

Sample Run:

```
Length of the first number set: 4

25 24 35 58

First set entered: 25 24 35 58

Length of the second number set:  4

Secon set entered: 5 6 45 29

Choose operation: +

30 30 80 87
```

```
Choose operation: -

20 18 -10 29

Choose operation: *

125 144 1575 1682

Choose operation: /

5 4 0.777778 2

Choose operation: exit
```

CENG 241 LAB 9

2- Write a C++ program that has 4 classes:

```
Car
-----------------------------------------
# brandname: string
# modelyear: int
# price: int
# totalCost: int
-----------------------------------------
Car(): Default const
Car(string,int,int): Overloaded const.
Car(&Car): Copy const
```

```
ElectricCar
-----------------------------------------
# NumberOfBatteries: int
# electricCost: int
-----------------------------------------
ElectricCar(): Default Const
ElectricCar(string,int,int,int,int): Overloaded Const
ElectricCar(&ElectricCar): Copy Const
friend ostream& operator<<(ostream &os, const
ElectricCar &obj): overloaded operator
```

```
GasolineCar
-----------------------------------------
# engineVolume: int
# fuelCost: int
-----------------------------------------
GasolineCar(): Default Const
GasolineCar(string,int,int,int,int)
GasolineCar(&GasolineCar): Copy Const
friend ostream& operator<<(ostream &os,
const Gasoline &obj): overloaded operator
```

```
Hybrid
-----------------------------------------

HybridCar(): Default Const
ElectricCar(&ElectricCar): Copy Const
friend ostream& operator<<(ostream &os, const
Hybrid &obj): overloaded operator
```

Car class is the parent class of “ElectricCar” and “GasolineCar” classes and “HybridCar” class is child of both “GasolineCar” and “ElectricCar” classes. You need to have getters and setters for each variables of all classes. You are required to have 3 vectors for “ElectricCar”, “GasolineCar” and “HybridCar” classes to store objects in your main function. Then, there should be a menu-driven process that let user add cars according to the type of the car. After user is done with the operations, program needs to print out all cars stored. You need to overload “<<” operator so that you may print your objects in an easier way (example: cout << obj). Overloaded operators must be defined with friend keyword. There are two type of costs varying according to the car type: fuel cost and electric cost. As hybrid cars have both fuel cost and electric cost, you need to calculate total cost considering this case.

Sample Output:

```
1-Electric Car 2- Gasoline Car 3- Hybrid Car 4-
Exit: 1

Enter brand name: BrandE

Enter model year: 2017

Price: 150000

Number Of Batteries: 2

Electric Cost: 20

1-Electric Car 2- Gasoline Car 3- Hybrid Car 4-
Exit: 2

Enter brand name: BrandG

Enter model year: 2019

Price: 120000

Engine Volume(CC): 1600

Fuel Cost: 100

1-Electric Car 2- Gasoline Car 3- Hybrid Car 4-
Exit: 3

Enter brand name: BrandH

Enter model year: 2017

Price: 220000

Number Of Batteries: 2

Electric Cost: 30

Engine Volume(CC): 1200

Fuel Cost: 100

1-Electric Car 2- Gasoline Car 3- Hybrid Car 4-
Exit: 3

Enter brand name: BrandH2

Enter model year: 2014

Price: 250000
```

```
Number Of Batteries: 3

Electric Cost: 20

Engine Volume(CC): 1400

Fuel Cost: 120

1-Electric Car 2- Gasoline Car 3- Hybrid Car 4-
Exit: 4

Electric Cars:

1-BrandE 2017 price: 150000 Number Of
batteries: 2 Electric Cost: 20 Total cost: 20

Gasoline Cars:

1-BrandG 2019 price: 120000 Engine Volume:
1600 Fuel Cost: 100 Total cost: 100

Hybrid Cars:

1-BrandH 2017 price: 220000 Number Of
Batteries: 2 Electric Cost: 30 Engine
Volume(CC): 1200 Fuel Cost: 100 Total Cost:
130

2- BrandH2 2014 price: 250000 Number Of
Batteries: 3 Electric Cost: 20 Engine
Volume(CC): 1400 Fuel Cost: 120 Total Cost:
150

BrandE has the minimum cost of all cars.
```