**Logo

Description automatically generated San Francisco Bay University**

**CS360L - Programming in C and C++ Lab**

**Lab Assignment #9**

**Due day: 07/26/2022**

**Instruction:**

1. **Push the answer sheets/source code to Github**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue lab assignment submission can’t be accepted.**

**4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Modify the vehicle management program to allow an automobile rental company to manage its fleet of automobiles. First, define a class called *CityCar* that contains an array of pointers to the *100* objects in the *Car* class. This also allows you to store pointers to objects of the derived class types *PassCar* and *Truck*. The objects themselves will be created dynamically at runtime.
   1. Define a class *CityCar* with an array of pointers to the *Car* class and an

*int* variable for the current number of elements in the array. The constructor will set the current number of array elements to *0*. The destructor must release memory allocated dynamically for the remaining objects. Make sure that you use a virtual destructor definition in the base class *Car* to allow correct releasing of memory for trucks and passenger vehicles.

Implement two versions of the *insert()* method using the prototype shown opposite. Each version will allocate memory to an object of the appropriate type—that is of the *PassCar* or *Truck* class—and use the arguments passed to it for initialization. The method should return *false* if it is impossible to enter another automobile (that is, if the array is full), and *true* in all other cases.

The *display()* method outputs the data of all vehicles on screen. To

perform this task it calls the existing *display()* method for each object.

* 1. Create a new function called *menu()* and store this function in a new source file. The function will display the menu shown opposite, read, and return the user’s choice.
  2. Additionally, write two functions, *getPassCar()* and *getTruck()*, which read the data for a car or a truck from the keyboard and write the data into the appropriate arguments.
  3. Create an object of the *CityCar* type in your *main* function. Insert one car and one truck. These will be the first vehicles of the company’s fleet. If a user chooses “Add car” or “Add truck,” your program must read the data supplied and call the appropriate version of *insert().*

// Definition of baseclass Car and

// of the derived class PassCar

// --------------------------------------------------

#include <iostream>

#include <string>

using namespace std;

class Car // Base class{

private:

long nr; //identification number

string producer;

public: // Constructor:

Car( long n = 0L, const string& prod = "");

// Access methods:

long getNr(void) const { return nr; }

void setNr( long n ) { nr = n; }

const string& getProd() const{ return producer; }

void setProd(const string& p){ producer = p; }

void display( void ) const; // Display a car

};

class PassCar : public Car // Derived class{

private:

string passCarType;

bool sunRoof;

public: // Constructor:

PassCar( const string& tp, bool sd,int n = 0 , const string& h = "");

// Access methods:

const string& getType() const{ return passCarType; }

void setType( const string s) { passCarType = s; }

bool getSunRoof() const { return sunRoof; }

void setSunRoof( bool b ) { sunRoof = b; }

void display() const;

};

class Truck : public Car {

private:

int axles;

double tons;

public:

Truck( int a, double t, int n, const string& hs);

~Truck();

void setAxles(int l){ axles = l;}

int getAxles() const { return axles; }

void setCapacity( double t) { tons = t;}

double getCapacity() const { return tons; }

void display() const;

};

1. An automatic checkout system for a supermarket chain needs to be completed.
   1. Declare the virtual methods *scanner()* and *printer()* in the base class *Product*. Also define a virtual destructor.
   2. Write the *record()* function, which registers and lists products purchased in the store in a program loop.

The function creates an array of *100* pointers to the base class, *Product*. The checkout assistant is prompted to state whether a prepacked or fresh food item is to be scanned next. Memory for each product scanned is allocated dynamically and referenced by the next pointer in the array. After scanning all the available items, a sequential list is displayed. The prices of all the items are added and the total is output at the end.

* 1. Now create an application program to simulate a supermarket checkout. The checkout assistant is prompted in a loop to state whether to define a new customer. If so, the *record()* function is called; if not, the program terminates.

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Product{

private:

long bar;

string name;

public:

Product(long b = 0L, const string& s = ""): bar(b), name(s){ }

// Access methods as previously used.

virtual void scanner(); // Virtual now!

virtual void printer() const;

};

class PrepackedFood : public Product{

private:

double pce\_price;

public:

PrepackedFood(double p = 0.0,long b = 0L, const string& s = ""): Product(b, s), pce\_price(p) {}

void setPrice(double p){ pce\_price = p;}

double getPrice()const { return pce\_price; }

void scanner(){

Product::scanner();

cout << "Price per piece: "; cin >> pce\_price;

}

void printer() const{

Product::printer();

cout << fixed << setprecision(2)

<< "Price per piece: " << pce\_price << endl;

}

};

class FreshFood : public Product{

private:

double wght;

double lbs\_price;

public:

FreshFood(double g = 0.0, double p = 0.0,long b = 0L, const string& s = ""): Product(b, s), wght(g), lbs\_price(p) {}

void setWght(double g) { wght = g;}

double getWght()const { return wght; }

void setPrice(double p) { lbs\_price = p;}

double getPrice()const { return lbs\_price; }

void scanner(){

Product::scanner();

cout << "Weight(lbs): "; cin >> wght;

cout << "Price/lbs: "; cin >> lbs\_price;

cin.sync(); cin.clear();

}

void printer() const{

Product::printer();

cout << fixed << setprecision(2)

<< "Price per Lbs: " << lbs\_price

<< "\nWeight: " << wght

<< "\nTotal: " << lbs\_price \* wght

<< endl;

}

};