TASK 1:

#include<iostream>

#include<string>

using namespace std;

class Time{

private:

int hr;

int min;

public:

Time(){

hr = 0;

min = 0;

}

Time(int hrs, int min){

hr = hrs;

min = min;

}

void setTime(int h, int m){

hr = h;

min = m;

}

void getTime(int, int){

cout << "HOUR : " << hr << endl;

cout << "MINUTE : " << min << endl;

}

void printTime(){

getTime(hr, min);

}

~Time(){

cout << "DESTRUCTOR OF TIME IS CALLED " <<endl;

}

};

class Date{

private:

int month;

int day;

int year;

public:

Date(){

month = 0;

day = 0;

year = 0;

}

Date(int m, int d, int y){

month = m;

day = d;

year = y;

}

void setDate(int m, int d, int y){

day = d;

month = m;

year = y;

}

void getDate(int, int&, int&){

cout << "DAY: " << day << endl;

cout << "MONTH: " << month << endl;

cout << "YEAR: " << year << endl;

}

void printDate(){

getDate(day, month, year);

}

~Date(){

cout << " DESTRUCTOR OF DATE IS CALLED" << endl;

}

};

class Event{

private:

string eventName;

Time eventTime;

Date eventDay;

public:

Event(int hours = 0, int minutes = 0, int m = 1,

int d = 1, int y = 1900, string name = "SPRING")

{

eventTime.setTime(hours, minutes);

eventDay.setDate(d, m, y);

eventName = name;

}

void setEventData(int hours, int minutes, int m, int d, int y, std::string name){

eventName = name;

}

void printEventData(){

cout << "EVENT NAME : " << eventName << endl;

eventDay.printDate();

cout << endl;

eventTime.printTime();

cout << endl;

}

~Event(){

cout << "destructor is called" <<endl;

}

};

int main(){

int hr, min, month, day, year;

string name;

cout << "INPUT EVENT NAME : " << endl;

getline(cin, name);

cin.ignore();

cout << "INPUT EVENT DAY : " << endl;

cin >> day;

cout << "INPUT EVENT MONTH : " << endl;

cin >> month;

cout << "INPUT EVENT YEAR : " << endl;

cin >> year;

cout << "INPUT EVENT HOUR : " << endl;

cin >> hr;

cout << "INPUT EVENT MINUTE : " << endl;

cin >> min;

Event obj(hr, min, day, month, year, name);

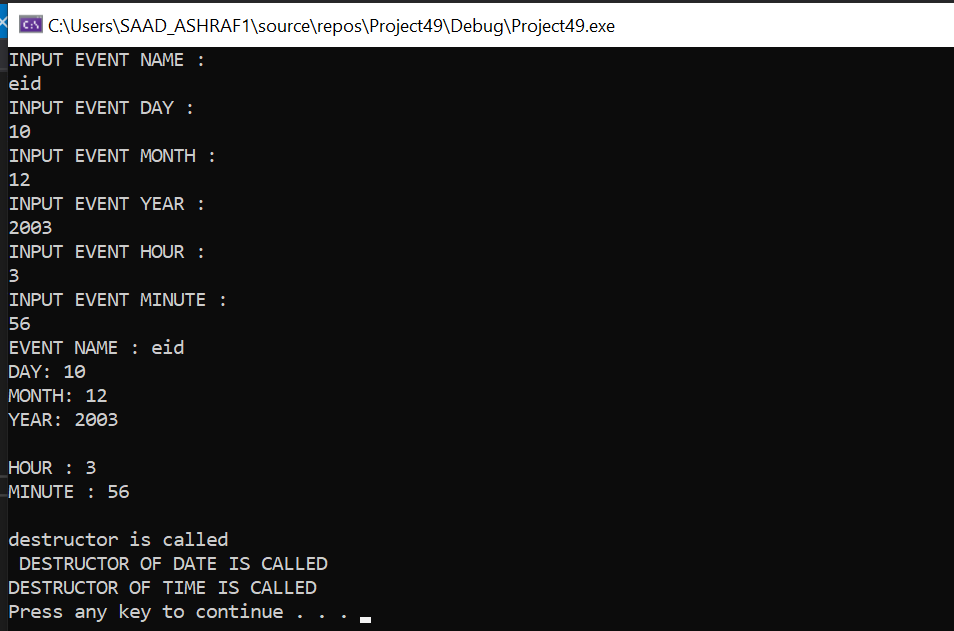
obj.printEventData();

obj.~Event();

system("pause");

return 0;

}



Task 2:

#include<iostream>

#include<string>

using namespace std;

class Car{

string car;

public:

Car(){

car = "";

}

Car(string c){

car = c;

}

void setCar(string c){

car = c;

}

string getCar(){

return car;

}

void DisplayCar(){

cout << "car = " << car <<endl;

}

void wheels(){

cout << "wheels turn forward and left and wheels are big" << endl;

}

void windows(){

string color;

cout << "Enter window color : " << endl;

cin >> color;

cout << "windows are " << color << " color " << endl;

}

void engine(){

int cc;

cout << "Enter cc of engine : "<< endl;

cin >> cc;

cout << "engine CC "<< cc << endl;

}

void door(){

cout << "door are open" <<endl;

}

void tires(){

cout << "car has four tires" << endl;

}

~Car(){

cout << "destructor" << endl;

}

};

class death{

private:

Car obj;

public:

void crash(){

cout << "car has crashed" << endl;

cout << "doors have been destroyed and all tires penctured" << endl;

cout << "engine is completely burned by fire hazard" << endl;

obj.~Car();

}

};

int main(){

string car;

cout << "enter car name : " << endl;

cin >> car;

Car obj1;

obj1.setCar(car);

obj1.getCar();

obj1.DisplayCar();

obj1.engine();

obj1.door();

obj1.windows();

obj1.wheels();

obj1.tires();

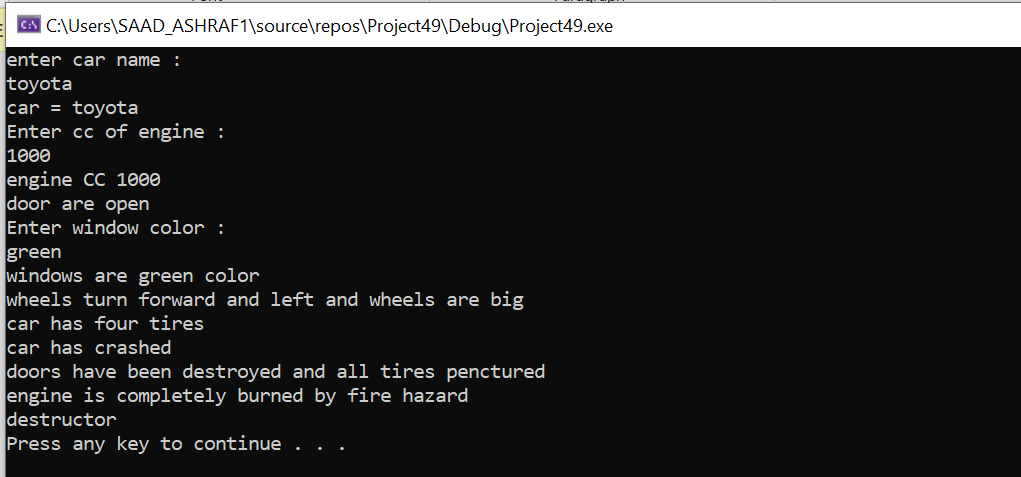
death obj;

obj.crash();

system("pause");

return 0;

}



TASK 3:

#include<iostream>

#include<string>

using namespace std;

class Address

{

public:

Address(string street, string town, string city){

this->street = street;

this->town = town;

this->city = city;

}

string retStreet(){

return street;

}

string retTown(){

return town;

}

string retCity(){

return city;

}

private:

string street, town, city;

};

class Person{

public:

Person(string name, Address\* address){

this->name = name;

this->address = address;

}

void Display(){

cout << "Name : " << name << endl;

cout << "Street : " << address->retStreet() << endl;

cout << "Town : " << address->retTown() << endl;

cout << "City : " << address->retCity() << endl;

cout << endl;

}

private:

string name, email;

Address\* address;

};

int main(){

string street, colony, city;

cout << "Enter street : " << endl;

cin >> street;

cout << "Enter colony : " << endl;

cin >> colony;

cout << "Enter city : " << endl;

cin >> city;

Address a1(street, colony, city);

Person obj1("Saad", &a1);

obj1.Display();

Person obj2("Ashraf", &a1);

obj2.Display();

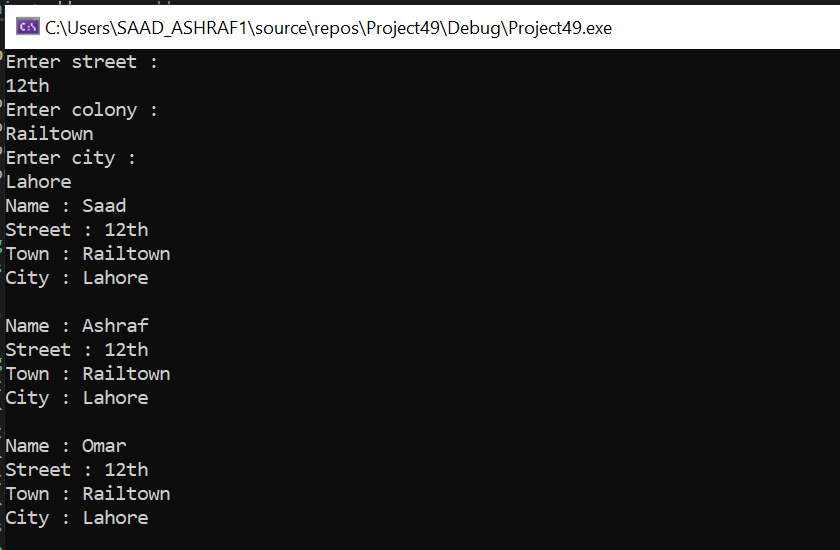
Person obj3("Omar", &a1);

obj3.Display();

system("pause");

return 0;

}



TASK 4:

#include<iostream>

#include<string>

using namespace std;

class Person{

private:

string name;

public:

Person(){

name = "";

}

Person(string n){

name = n;

}

void setter(string n){

name = n;

}

string getter(){

return name;

}

void print(){

cout << name <<endl;

}

~Person(){

cout << "person destructor has been called" << endl;

}

};

class Teacher{

private:

string subject;

Person\* obj;

public:

Teacher(){

subject = "";

}

Teacher(string s, Person\* ptr){

subject = s;

obj = ptr;

}

void setter1(std::string s, Person\* ptr){

subject = s;

obj = ptr;

}

string getter1() {

return subject;

}

void print1(){

cout << subject << endl;

obj->print();

cout << endl;

}

~Teacher(){

cout << "teacher destructor has been called" << endl;

}

};

int main(){

string name;

cout << "enter name " << endl;

cin >> name;

Person obj(name);

Teacher obj2("math", &obj);

obj2.print1();

obj2.~Teacher();

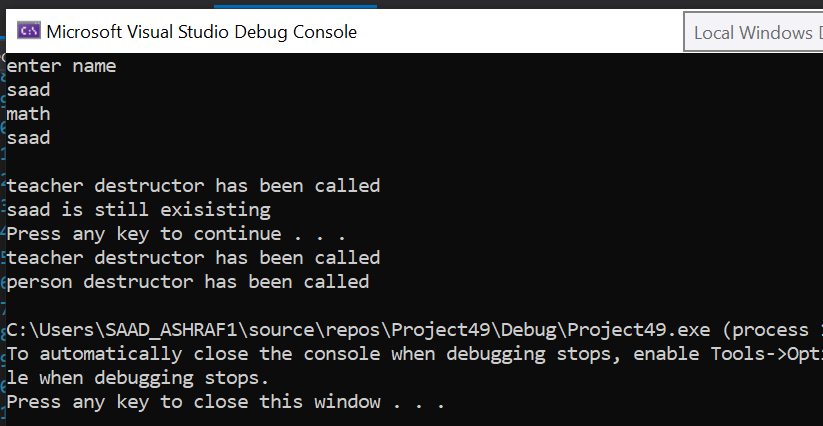
cout << obj.getter();

cout << " is still exisisting" << endl;

system("pause");

return 0;

}



TASK 6:

#include <iostream>

using namespace std;

class box {

private:

int length;

int breadth;

int height;

public:

void set\_dimensions(int l, int b, int h){

length = l;

breadth = b;

height = h;

}

void show\_data(){

cout << " Length : " << length << endl << " Breadth : " << breadth << endl << "Height : " << height << endl;

}

};

int main()

{

box B1, B3;

B1.set\_dimensions(30, 30, 30);

B1.show\_data();

box B2 = B1; // COPY CONSTRUCTOR

B2.show\_data();

// copy is done by default

B3 = B1;

B3.show\_data();

return 0;

}

