DSA Lab01

23K2001

M.Muzammil Siddiqui

BCS-3J

Q1:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

class bankAccount{

    private:

    float balance;

    public:

    bankAccount() { balance = 0.0; }

    bankAccount(float b){ balance = b; }

    float display(){ return balance; }

    void deduct(float x){ balance-=x; }

};

int main(){

    bankAccount account1;

    cout<<"Acc#1: "<<account1.display()<<endl;

    bankAccount account2(1000);

    cout<<"Acc#2: "<<account2.display()<<endl;

    bankAccount account3(account2);

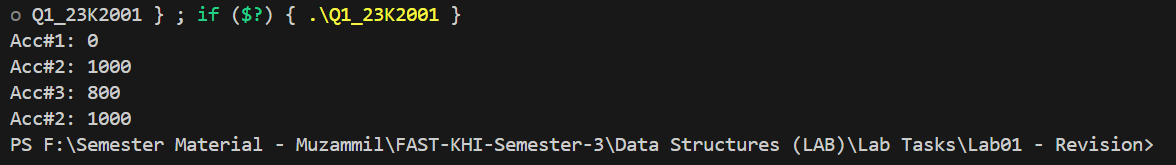
    account3.deduct(200);

    cout<<"Acc#3: "<<account3.display()<<endl;

    cout<<"Acc#2: "<<account2.display()<<endl;

    return 0;

}



Q2:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

class Exam{

    private:

    string name,date;

    char score;

    public:

    void setRecord(string n,string d,char s){ name=n;

    date = d;

    score = s;

    }

    void examDetails(){

        cout<<"Name: "<<name<<endl;

        cout<<"Date: "<<date<<endl;

        cout<<"Score: "<<score<<endl;

    }

};

int main(){

    Exam e1;

    e1.setRecord("Muzammil","23-Aug-2024",'A');

    e1.examDetails();

    cout<<endl;

    Exam e2 = e1;

    e2.examDetails();

    cout<<endl;

    e1.setRecord("Muzammil","27-Feb-2024",'B');

    e1.examDetails();

    cout<<endl;

    e2.examDetails();

    return 0;

}

//If we do not implement our own copy constructor then the objects will use the same memory

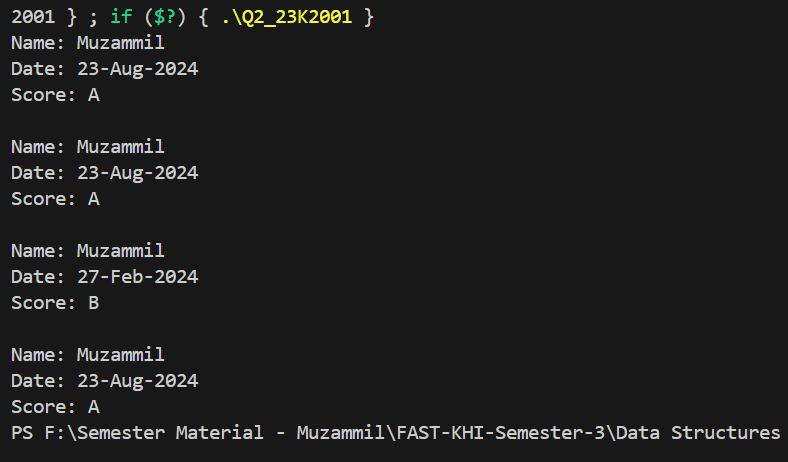
//because of this issue, when object destructor is called, it will destruct the same object

//twice that can problem with program. If we change some record then it does not reflect in the copied object

//This was due to shallow copy constructor, it can be solved by making deep copy constructor that allocates separate memory

//for the new object that is created. So no issues will be encountered in pointers and dynamic attributes however

//in our question there are no dynamic memories..



Q3:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

class Document{

    private:

    char \*text;

    int length;

    public:

    Document(){ text = NULL;

    length = 0;}

    Document(string t){ length = t.size();

        text = new char[length];

        for(int i=0;i<length;i++)

        text[i] = t[i];

    }

    Document(Document &obj) {

    this->length = obj.length;

    this->text = new char[this->length];

    for (int i=0; i<this->length;i++)

        this->text[i] = obj.text[i];

    }

    void display(){

        for(int i=0;i<length;i++)

        cout<<text[i];

        cout<<endl;

    }

    void replace(string t){

        delete[] this->text;

        length = t.size();

        text = new char[length];

        for(int i=0;i<length;i++)

        text[i] = t[i];

     }

    Document& operator=(Document &obj) {

    delete[] text;

    length = obj.length;

    text = new char[length];

    for (int i=0; i<length; i++)

        text[i] = obj.text[i];

    return \*this;

}

    ~Document(){ delete[] text;}

};

int main(){

    char s1[] = "This is an important file.";

    Document file1(s1);

    cout<<"File1: ";

    file1.display();

    Document file2(file1);

    Document file3;

    file3 = file1;

    cout<<"File2: ";

    file2.display();

    cout<<"File3: ";

    file3.display();

    char s2[] = "File has been edited.";

    file1.replace(s2);

    cout<<"File1: ";

    file1.display();

    cout<<"File2: ";

    file2.display();

    cout<<"File3: ";

    file3.display();

    return 0;

}

