LAB:3

Sample program:

*To study the initialization of data members through constructors.

Code:

```
#include<iostream>
using namespace std;
class Distance {
    private:
        int mts;
        int cms;
    public:
        Distance(){
            mts=0;
            cms=0;
            cout<<"constructure is called"<<endl;
        }
        void get_data();
        void display();
        void cal_distance(Distance,Distance);
};
void Distance::get_data(){
    cout<<"enter Distance in mts"<<endl;
    cin>>mts;
    cout<<"enter data in cms"<<endl;
    cin>>cms;
}
void Distance::display(){
    while((cms>=10)|(cms>0 & mts<0)){
        mts++;
        cms-=100;
    }
    while(cms<0 & mts>0){
        mts--;
        cms+=100;
```

```
    }
    cout<<"meters: "<<mts<<endl<<"centimeters :"<<cms<<endl;
}
void Distance::cal_distance(Distance d1,Distance d2){
    mts=d1.mts-d2.mts;
    cms=d1.cms-d2.cms;
    cout<<cms<<endl<<mts<<endl;
}
int main(){
    Distance di1,di2,di3;
    di1.get_data();
    di2.get_data();
    di3.cal_distance(di1,di2);
    cout<<"content of di1: "<<endl;
    di1.display();
    cout<<"content of di2: "<<endl;
    di2.display();
    cout<<"content of di3: "<<endl;
    di3.display();
    return (0);
}
```

Output:

```
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3 60x25
kashyap@kash  ~/cpp/LAB_3  g++ sample_proj.cpp
kashyap@kash  ~/cpp/LAB_3  ./a.out
constructure is called
constructure is called
constructure is called
enter Distance in mts
1
enter data in cms
200
enter Distance in mts
1
enter data in cms
100
100
0
content of di1:
meters: 3
centimeters :0
content of di2:
meters: 2
centimeters :0
content of di3:
meters: 1
centimeters :0
kashyap@kash  ~/cpp/LAB_3
```

1st modification:

***Modify the above program to overload the Constructor to initialize new objects by passing parameters in two ways :**

a) Distance in mts and cms is passed.

b) Only distance in mts is passed.

Code:

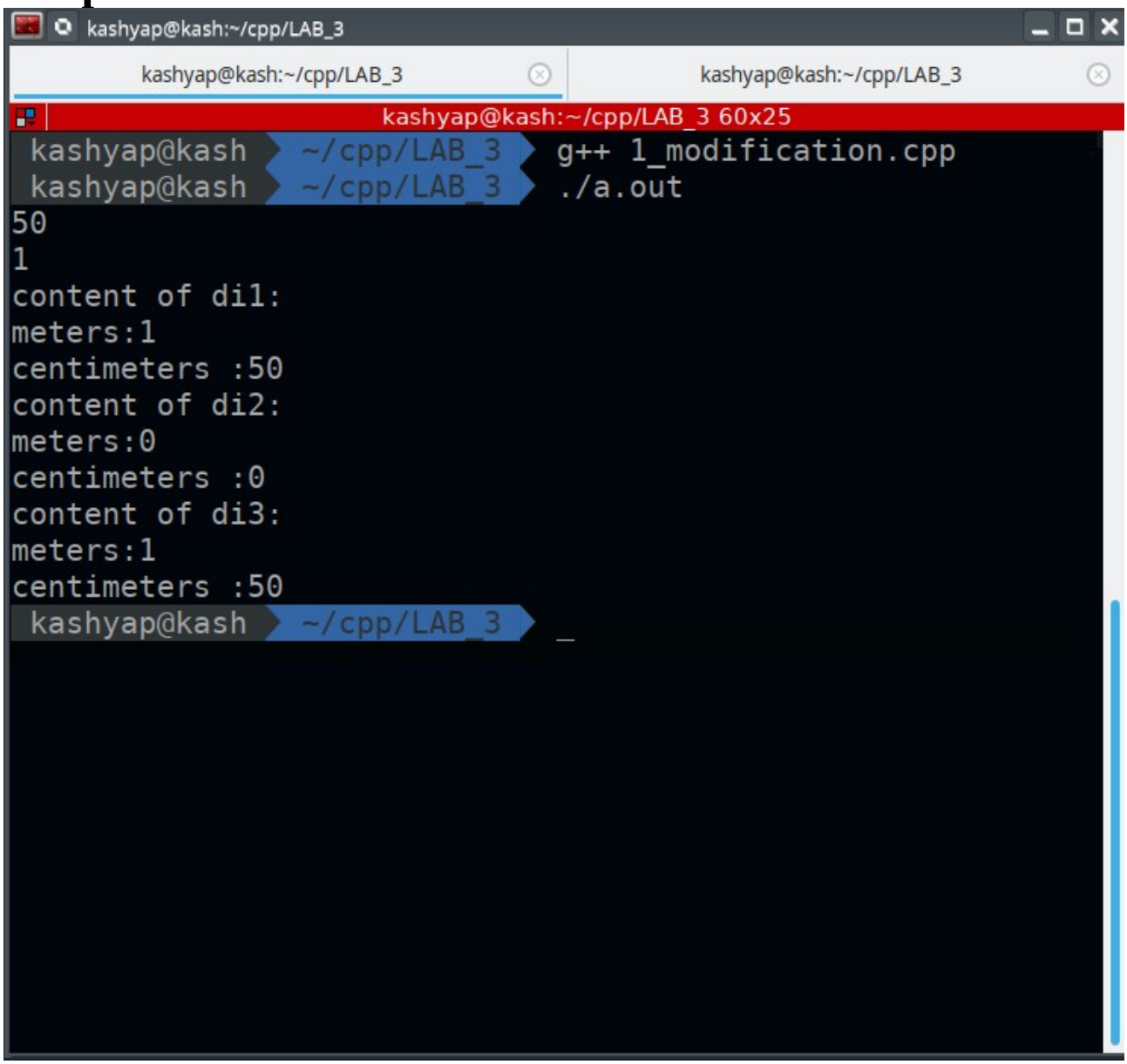
```
#include<iostream>
using namespace std;
int cnt1=0;
class Distance {
    private:
        int mts;
        int cms;
    public:
        //constructure with no argument
        Distance(){
            mts=0;
            cms=0;
            // cout<<"constructure is called"<<endl;
        }
        //constructue is having 2 arguments
        Distance(int buff1,int buff2){
            mts=buff1;
            cms=buff2;
            // cout<<"constructure is called"<<endl;
        }
        //constructur with 1 argument
        Distance(int buff2){
            mts=buff2;
            cms=0;
            // cout<<"constructure is called"<<endl;
```

```
        }
        void get_data();
        void display();
        void cal_distance(Distance,Distance);
};
void Distance::get_data(){
    cout<<"enter Distance in mts"<<endl;
    cin>>mts;
    cout<<"enter data in cms"<<endl;
    cin>>cms;
}
void Distance::display(){
    while((cms>=10)|(cms>0 & mts<0)){
        mts++;
        cms-=100;
    }
    while(cms<0 & mts>0){
        mts--;
        cms+=100;
    }
    cout<<"meters:"<<mts<<endl<<"centimeters :"<<cms<<endl;
}
void Distance::cal_distance(Distance d1,Distance d2){
    mts=d1.mts-d2.mts;
    cms=d1.cms-d2.cms;
    cout<<cms<<endl<<mts<<endl;
}
int main(){
    Distance di1(1,50),di2(0),di3;
    // di1.get_data();
    // di2.get_data();
    di3.cal_distance(di1,di2);
    cout<<"content of di1: "<<endl;
    di1.display();
    cout<<"content of di2: "<<endl;
```

```
    di2.display();
    cout<<"content of di3: "<<endl;
    di3.display();
    return (0);

}
```

Output:



2nd modification:

***Remove GetData( ) if it has become redundant.**
(its as same as 1st modification)

Code:

```
#include<iostream>
using namespace std;
int cnt1=0;
class Distance {
    private:
        int mts;
        int cms;
    public:
        //constructure with no argument
        Distance(){
            mts=0;
            cms=0;
            // cout<<"constructure is called"<<endl;
        }
        //constructue is having 2 arguments
        Distance(int buff1,int buff2){
            mts=buff1;
            cms=buff2;
            // cout<<"constructure is called"<<endl;
        }
        //constructur with 1 argument
        Distance(int buff2){
            mts=buff2;
            cms=0;
            // cout<<"constructure is called"<<endl;
        }
        void display();
```

```
        void cal_distance(Distance,Distance);
};

void Distance::display(){
    while((cms>=10)|(cms>0 & mts<0)){
        mts++;
        cms-=100;
    }
    while(cms<0 & mts>0){
        mts--;
        cms+=100;
    }
    cout<<"meters:"<<mts<<endl<<"centimeters :"<<cms<<endl;
}
void Distance::cal_distance(Distance d1,Distance d2){
    mts=d1.mts-d2.mts;
    cms=d1.cms-d2.cms;
    cout<<cms<<endl<<mts<<endl;
}
int main(){
    Distance di1(1,50),di2(0),di3;

    di3.cal_distance(di1,di2);
    cout<<"content of di1: "<<endl;
    di1.display();
    cout<<"content of di2: "<<endl;
    di2.display();
    cout<<"content of di3: "<<endl;
    di3.display();
    return (0);

}
```

Output:

```
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3 60x25
kashyap@kash  ~/cpp/LAB_3  g++ 2_modification.cpp
kashyap@kash  ~/cpp/LAB_3  ./a.out
50
1
content of di1:
meters:1
centimeters :50
content of di2:
meters:0
centimeters :0
content of di3:
meters:1
centimeters :50
kashyap@kash  ~/cpp/LAB_3  _
```

3rd modification:

***Modify each Constructor and Destructor such that it also display the number of times it is called.**

Code:

```
#include<iostream>
using namespace std;
static int cnt =0;
static int cnt1=0;
class Distance {
    private:
        int mts;
        int cms;
    public:
        Distance(){
            mts=0;
            cms=0;
            cnt++;
            cout<<"constructor is called "<<cnt<<" times "<<endl;
        }
        ~Distance(){
            cnt1++;
            cout<<"destructor is called "<<cnt1<<" times "<<endl;
        }
        void get_data();
        void display();
        void cal_distance(Distance,Distance);
};
void Distance::get_data(){
    cout<<"enter Distance in mts"<<endl;
    cin>>mts;
    cout<<"enter data in cms"<<endl;
    cin>>cms;
```

```
}
void Distance::display(){
    while((cms>=10)|(cms>0 & mts<0)){
        mts++;
        cms-=100;
    }
    while(cms<0 & mts>0){
        mts--;
        cms+=100;
    }
    cout<<"meters:"<<mts<<endl<<"centimeters :"<<cms<<endl;
}
void Distance::cal_distance(Distance d1,Distance d2){
    mts=d1.mts-d2.mts;
    cms=d1.cms-d2.cms;
    cout<<cms<<endl<<mts<<endl;
}
int main(){
    Distance di1,di2,di3;
    di1.get_data();
    di2.get_data();
    di3.cal_distance(di1,di2);
    // cout<<"content of di1: "<<endl;
    // di1.display();
    // cout<<"content of di2: "<<endl;
    // di2.display();
    // cout<<"content of di3: "<<endl;
    // di3.display();
    return (0);

}
```

Output:

```
kashyap@kash:~/cpp/LAB_3
kashyap@kash:~/cpp/LAB_3 60x30
kashyap@kash  ~/cpp/LAB_3  g++ 3_modification.cpp
kashyap@kash  ~/cpp/LAB_3  ./a.out
constructor is called 1 times
constructor is called 2 times
constructor is called 3 times
enter Distance in mts
1
enter data in cms
11
enter Distance in mts
1
enter data in cms
1
10
0
destructor is called 1 times
destructor is called 2 times
destructor is called 3 times
destructor is called 4 times
destructor is called 5 times
kashyap@kash  ~/cpp/LAB_3  _
```