OOP LAB 11

TOTAL MARKS: 80

THE OBJECTIVE OF THIS LAB IS TO:

1. Understand and practice inheritance on singular and multi-level.
2. Understand and practice function overloading.
3. Practice good coding conventions e.g commenting, meaningful variable and functions names, properly indented and modular code.

INSTRUCTIONS!

1. This is a graded lab, you are strictly NOT allowed to discuss your solutions with your fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me.
2. Strictly follow good coding conventions (commenting, meaningful variable and functions names, properly indented and modular code.
3. Save your work frequently. Make a habit of pressing CTRL+S after every line of code you write.

> "STOP ACTING SO SMALL. YOU ARE THE UNIVERSE IN ECSTATIC MOTION. WHAT YOU SEEK IS SEEKING YOU"
>
> - RUMI

CONSIDER AND IMPLEMENT THE FOLLOWING FILE DEFINITIONS:

Pair.h

MARKS: 5

```
class Pair {
public:
    CString key;
    CString value;
    Pair(const CString & key, const CString * value);
    ~Pair();
};
```

PairArray.h

MARKS:10

```
class PairArray
{
    int size;
    Pair * pairs;
    int findKey(const CString & key); //returns index of a key
    void resize(int newSize);
    bool isFull(int count);
    bool isEmpty(int count);
public:
    PairArray(int size);
    ~PairArray();
    void insertData(const CString value, const CString key);
       //inserts key to a specific value
    CString extractData(const CString key);
       //returns value of a specific key
};
```

JSON.h

MARKS: 35

```
class JSON : public PairArray
{
    DataConverter converter;
    int count;
    //Represents a JSON object
public:
    JSON(int size);
    ~JSON();

    void insert(int value,CString key){
        CString converted;
        converter.dataConverter(value, converted);
        this->insertData(converted, key);
        count++;
    } //inserts Integer
    void insert(CString,CString key); //inserts CString
    void insert(float,CString key); //inserts float
    void insert(Date,CString key); //inserts Date

    void getData(CString &, const CString & key);   //returns CString
    void getData(Date &, const CString & key);      //returns Date
    void getData(int & value, const CString & key) {
        CString extracted = this->extractData(key);
        converter.dataConverter(extracted,value);
    }       //returns int
    void getData(float &, const CString & key);     //returns float
    int getCount();
};
```

Document.h

MARKS: 10

```
class Document: public JSON
{
    //Represents a Document object
public:
    Document(int size);
    ~Document();
    void display();
    //SAMPLE DISPLAY:
    /*
    Document =
    {
        "name" : "Ansab Gillani"
        "age" : 22
        "roll number" : "BCSF17M047"
        "dob" : "13th May 1999"
        "cgpa" : 3.23
    }
    */
    void displayValue();
    void input();  //Takes a key and a value and stores it
};
```

DataConverter.h

MARKS: 20

```
class DataConverter
{
public:
    void dataConverter(CString & destination,const Date & source);
       //Date to CString
    void dataConverter(Date &,const CString &);     //CString to  Date
    void dataConverter(CString &, const int);       //Int to CString
    void dataConverter(int & destination, const CString source){
        destination = source.toInteger();
    }             //CString to Int
    void dataConverter(CString &, const float);     //Float to CString
    void dataConverter(float &, const CString);           //CString to Float
};
```

Bonus

MARKS: 30

Implement Sub-Documents inside Documents:

```
    //SAMPLE DISPLAY:
    /*
    Document =
    {
        "name" : "Ansab Gillani"
        "age" : 22
        "roll number" : "BCSF17M047"
        "dob" : "13th May 1999"
        "cgpa" : 3.23
        "address" : {
                "city" : "Lahore"
                "town" : "Askari 11"
                "country" : "Pakistan"
                }
    }
    */
```

Date.h

For those who have not yet successfully implemented Date/Time Class, here is an implementation of my Date class.

```
enum Months { JANUARY = 31,
                FEBURARY = 28,
                MARCH = 31,
                APRIL = 30,
                MAY = 31,
                JUNE = 30,
                JULY = 31,
                AUGUST = 31,
                SEPTEMBER = 30,
                OCTOBER = 31,
                NOVEMBER = 30,
                DECEMBER = 31
};
class Date
{
private:
    int year;
    int month;
    int day;
    bool isLeapYear(int y);
    int getNumberOfDaysOfMonths(int m, int y);
    int getTotalNumberOfDaysOfYear();
public:
    Date();
    void setYear(int y);
    void setMonth(int m);
    void setDay(int d);
    void setDate(int y, int m, int d);
    void display();
    int getYear();
    int getMonth();
    int getDay();
```

```
    void incDay(int d = 1);
    void incMonth(int m = 1);
    void incYear(int y = 1);
    void decDay(int d = 1);
    void decMonth(int m = 1);
    void decYear(int y = 1);
};
```

Date.cpp

```cpp
#include<iostream>
#include<iomanip>
#include"Date.h"
using namespace std;
Date::Date()
{
    month = 1;
    year = 1600;
    day = 1;
}
void Date::setMonth(int m)
{
    if (m >= 1 && m <= 12)
        month = m;
}
void Date::setYear(int y)
{
    if (y >= 1000)
        year = y;
}
void Date::setDay(int d)
{
    if (d >= 1 && d <= 31)
    {
        if (d <= 31 && month == 1 || month == 3 || month == 5 || month == 7 ||
 month == 8 || month == 10 || month == 12)
            day = d;
        else if (d <= 30 && month == 4 || month == 6 || month == 9 || month ==
 11)
            day = d;
        else if (isLeapYear(year) && d <= 29 && month == 2)
            day = d;
        else if (!isLeapYear(year) && d <= 28 && month == 2)
            day = d;
    }
```

```
}
void Date::setDate(int d, int m, int y)
{
    setYear(y);
    setMonth(m);
    setDay(d);
}
void Date::display()
{
    int day = getDay(), m = getMonth(), y = getYear();
    cout << setfill('0') << right << setw(2) << day << " / " << setfill('0') <
< right << setw(2) << m << " / " << setfill('0') << right << setw(4) << y;
}
bool Date::isLeapYear(int y)
{
    if (y % 4 == 0 )
        return true;
    return false;
}
int Date::getYear()
{
    return year;
}
int Date::getMonth()
{
    return month;
}
int Date::getDay()
{
    return day;
}
int Date::getNumberOfDaysOfMonths(int m, int y)
{
    if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
        return 31;
    else if (m == 4 || m == 6 || m == 9 || m == 11)
```

```
        return 30;
    else if (isLeapYear(y) && m == 2)
        return 29;
    else if (!isLeapYear(y) && m == 2)
        return 28;
}
int Date::getTotalNumberOfDaysOfYear()
{
    int totalDays = 0;
    for (int i = 1; i < month; i++)
    {
        totalDays = totalDays + getNumberOfDaysOfMonths(i, year);
    }
    totalDays = totalDays + day;
    return totalDays;
}
void Date::incDay(int d)
{
    if (d >= 0)
    {
        if (day + d > 0 && day + d <= getNumberOfDaysOfMonths(month, year))
            day = day + d;
        else
        {
            long long tDays = getTotalNumberOfDaysOfYear();
            tDays = tDays + d;
            while (isLeapYear(year) && tDays > 366 || !isLeapYear(year) && tDa
ys > 365)
            {
                if (isLeapYear(year))
                {
                    tDays = tDays - 366;
                    incYear();
                }
                else
                {
```

```
                    tDays = tDays - 365;
                    incYear();
                }
            }
            month = 1;
            while (tDays > getNumberOfDaysOfMonths(month, year))
            {
                tDays = tDays - getNumberOfDaysOfMonths(month, year);
                incMonth();
            }
            day = (int)tDays;
        }
    }
}
void Date::incMonth(int m)
{
    if (m >= 0)
    {
        int oldDay = getNumberOfDaysOfMonths(month, year);
        if (m % 12 == 0)
            incYear(m / 12);
        else if ((month + m) % 12 == 0)
        {
            incYear((month + m) / 12);
            month = 1;
        }
        else if (month + m > 12)
        {
            incYear((month + m) / 12);
            month = month + m;
            if (month % 12 == 0)
                month = 1;
            else
                month = month % 12;
        }
        else
```

```
            month = month + m;
        if (day == oldDay && day > getNumberOfDaysOfMonths(month, year))
            day = getNumberOfDaysOfMonths(month, year);
    }
    else
        return;
}
void Date::incYear(int y)
{
    if (y >= 0)
    {
        if (day == getNumberOfDaysOfMonths(month, year))
        {
            if (day > getNumberOfDaysOfMonths(month, year + y))
                day = getNumberOfDaysOfMonths(month, year + y);
        }
        year = year + y;
    }
    else
        return;
}
void Date::decDay(int d)
{
    if (d >= 0)
    {
        if (day - d > 0)
            day = day - d;
        else
        {
            long long tDays = getTotalNumberOfDaysOfYear();
            if (tDays - d > 0)
            {
                tDays = tDays - d;
                month = 1;
                while (tDays > getNumberOfDaysOfMonths(month, year))
                {
```

```
                tDays = tDays - getNumberOfDaysOfMonths(month, year);
                incMonth();
            }
            day = (int)tDays;
        }
        else if (tDays - d == 0)
        {
            month = 12;
            day = 31;
            decYear();
        }
        else if (tDays - d < 0)
        {
            while (d > tDays)
            {
                d = d - tDays;
                decYear();
                month = 12;
                tDays = getTotalNumberOfDaysOfYear();
            }
            month = 12;
            int mDays = getNumberOfDaysOfMonths(month, year);
            while (d >= mDays)
            {
                d = d - mDays;
                decMonth();
                mDays = getNumberOfDaysOfMonths(month, year);
            }
            if (day==0)
                day = getNumberOfDaysOfMonths(month, year);
            else
                day = getNumberOfDaysOfMonths(month, year) - d;
        }
    }
}
```

```
void Date::decMonth(int m)
{
    if (m >= 0)
    {
        int oldDay = getNumberOfDaysOfMonths(month,year);
        if (m % 12 == 0)
            decYear(m / 12);
        else if (abs(month - m) % 12 == 0)
        {
            decYear(abs(month - m) / 12 + 1);
            month = 12;
        }
        else if (month - m < 0)
        {
            decYear(abs(month - m) / 12 + 1);
            month = month - m;
            month = 12 - abs(month) % 12;
        }
        else
            month = month - m;
        if (day == oldDay && day > getNumberOfDaysOfMonths(month, year))
            day = getNumberOfDaysOfMonths(month, year);
    }
    else
        return;
}
void Date::decYear(int y)
{
    if (y >= 0 && year - y >= 1000)
    {
        if (day == getNumberOfDaysOfMonths(month, year))
        {
            if (day > getNumberOfDaysOfMonths(month, year - y))
                day = getNumberOfDaysOfMonths(month, year - y);
        }
        year = year - y;
```

```
    }
    else
        return;
}
```