NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD

Object Oriented Programming Fall 2022

ASSIGNMENT # 02

Due Date:

23rd October, 2022 (Sunday)

Mark your calendar as this is a hard deadline

Instructions:

- Make sure that you read and understand each and every instruction
- Create each problem solution in a separate header file such as: 'Problem1.h' for problem 1
- Keep a backup of your work always that will be helpful in preventing any mishap.
- Combine all your work in one .zip file.
- Declare functions outline.
- You are being provided with a test.cpp containing test cases for all questions. Don't change anything in test.cpp unless instructed.
- There is no need to submit .cpp files except test.cpp
- Name the .zip file as ROLL-NUM SECTION.zip (e.g. 21i-0001 B.zip).
- Submit the .zip file on Google Classroom within the deadline.
- Avoid last hour submissions
- Start early otherwise you will struggle with the assignment.
- You must follow the submission instructions to the letter, as failing to do so will get you a zero in the assignment.
- All the submitted evaluation instruments will be checked for plagiarism. If found plagiarized, both the involved parties will be marked 0

Total Marks: 10

Q 1: class Invoice--Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four data members

char* partNum — that points to a "part number"
char* partDesc — that points to part description
int quantity — a quantity of the item being purchased
float pricePerItem — price of single item

Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.

Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class Invoice{
private:
    // think about the private data members...
public:
    // provide definitions of following functions...
    Invoice();// default constructor
    Invoice(char* partNum, char* partDesc, int quantity,float pricePerItem); // parameterized
    constructor that initializes all data members
    Invoice(Invoice &i); // copy constructor

    //implement mutators for all private data members
    //implement accessors for all private data members

    //you have to implement the following functions as well
    float getInvoiceAmount();
    ~Invoice();
};
```

Total Marks: 15

Q2: Date Class – Create a class called Date that includes three pieces of information as data members

int month

int day

int year

Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. Perform error checking on the initializer values for data members month, day and year. Ensure that the month value is in the range 1–12; if it isn't, set the month to 1. Provide a set and a get function for each data member. Provide a member function displayDate that displays the month, day and year separated by forward slashes (/).

Provide a member function nextDay to increment the day by one. Calling nextDay function on the last date of a month should increment the month as well. Similarly, calling nextDay function on the last date of a year should increment the year as well. Be sure to test the following cases: a) Incrementing into

the next month. b) Incrementing into the next year. While incrementing month consider the year value, if the year is a leap year the last day of feberary is 29 otherwise 28.

For finding leap year, follow the rules:

If the year is evenly divisible by 4, 100, and 400, it is a leap year (it has 366 days).
Otherwise the year is not a leap year (it has 365 days).

Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class Date{
private:
    // think about the private data members...
public:
    // provide definitions of following functions...
    Date(); // default constructor
    Date(int day, int month, int year);  // parameterized constructor that initializes all data members
     Date(Date &d);  // copy constructor

    //implement mutators for all private data members
    //implement accessors for all private data members

    //you have to implement the following functions
    void nextDay()
    ~Date();
};
```

Total Marks: 20

Q 3: DateAndTime Class -- Combine the features of Time and Date into one class called DateAndTime. In addition to data members of Date class the DateAndTime class has following data members:

int second
int minute
int hour
char meridiem – possible values ‘a’ or ‘p’

The 12-hour clock divides the 24-hour day into two periods. ‘am’ stands for the Latin ante meridiem, translating to "before midday". This is the time before the sun has crossed the meridian. ‘pm’ stands for post meridiem or "after midday" – after the sun has crossed the meridian.

In addition to other functions the class has a tick function which increments the time by 1 second. Calling tick function on the last second of a minute should increment the minute as well. Similarly, calling tick function on the last second of an hour should increment the hour as well.

The tick function should call the nextDay function if the time increments into the next day keeping in vie the value of meridiem. The function displayDate should output the date and time. Be sure to test incrementing the time into the next day.

Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class DateAndTime{
private:
    // think about the private data members...
public:
    // provide definitions of following functions...
    DateAndTime(); // default constructor
    DateAndTime(int day, int month, int year, int second, int minute, int hour, char meridiem);  //
    parameterized constructor that initializes all data members
     DateAndTime(DateAndTime &d);  // copy constructor

    //implement mutators for all private data members
    //implement accessors for all private data members

    //you have to implement the following functions
    void nextDay()
    void tick()
    ~ DateAndTime ();
};
```

Total Marks: 45

Q 4: Implementation of Integer Class – Your goal is to implement “Integer" class. Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class Integer{
  // think about the private data members...
  public:
```

```
    //include all the necessary checks before performing the operations in the functions
    Integer();// a default constructor
    Integer(int);// a parametrized constructor
    Integer(String); // a parametrized constructor
    void set(int);//set value
    int get()const;//get value at (i,j)
    int bitCount(); //Returns the number of one-bits in the 2's complement binary
    int compareTo(Integer); //Compares two Integer objects numerically.
    double doubleValue(); //Returns the value of this Integer as a double.
    float floatValue(); //Returns the value of this Integer as a float.
    Integer plus(const Integer&); //adds two Integers and return the result
    Integer minus(const Integer&); // subtracts two Integers and return the result
    Integer multiple(const Integer&); //multiplies two Integers and return the result
    Integer divide(const Integer&); //divides two Integers and return the result
    static int numberOfLeadingZeros(int i); /*Returns the number of zero bits preceding the highest-order
    ("leftmost") one-bit in the two's complement binary representation of the specified int value.*/
    static int numberOfTrailingZeros(int i); /*Returns the number of zero bits following the lowest-order
    ("rightmost") one-bit in the two's complement binary representation of the specified int value.*/
    static String toBinaryString(int i); //Returns string representation of i
    static String toHexString(int i); //Returns string representation of i in base16
    static String toOctString(int i); //Returns string representation of i in base 8
};
```

Total Marks: 45

Q5: Implementation of Family Tree –Create a class Person that should store following attributes of a person

string name --- name of person

char gender --- gender as ‘M’, ‘F’

Date dob --- an object of Date class

int noOfChildren – total no. of children registered in familyTree so far. Upon each registration this should be incremented by 1

public: Person **children=new Person*[3] – a double pointer pointing to array of pointers that point to each registered child. Assume that family tree can register 3 children per person at max.

```
class Person{
private:
    // think about the private data members...
public:
    // provide definitions of following functions...
    Person ();// a default constructor
```

```
    Person(string n, char g, int day, int month, int year);// a parametrized constructor

    //implement mutators for all private data members
    //implement accessors for all private data members

    //you have to implement the following functions
    void displayData()  // prints data members
    int calcAge()       // subtracts the date of birth from current date (can be taken from user) and
    display age in years
    ~Person(); // delete dynamic element
};
```

NADRA is a national database that keeps record of nationals. Family Registration Certificate (FRC) is a mean of being identified with your NADRA's record. This provides the family composition. You're required to design a FamilyTree class that keeps record of a person, his children, grandchildren and so on. Each FamilyTree must have an ancestor or forefather who got registered for the first time. The core responsibility of a familyTree class is to register 3 or less children of each person, displaying the data of whole tree , finding other facts about the family.

Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class FamilyTree {
private:
    Person* foreFather;
            // the first person of a
family
public:
    // provide definitions of following functions...
    FamilyTree (Person* foreFather);// no default constructor as tree always starts from an ancestor

    //implement mutators for all private data members
    //implement accessors for all private data members

    //you have to implement the following functions
    void registerChild(Person &p, Person &child)  // a child is always registered against a parent. This
    function will add an element child to the children array in person p and update the total no of
    children of person accordingly

    Person findYoungestChildOf(Person &p)   // a youngest child is the one with lowest age among
    siblings. This should call calculate age function to get age of each child. For example the youngest
    child of Robert is Alice as shown in a sample tree below.
```

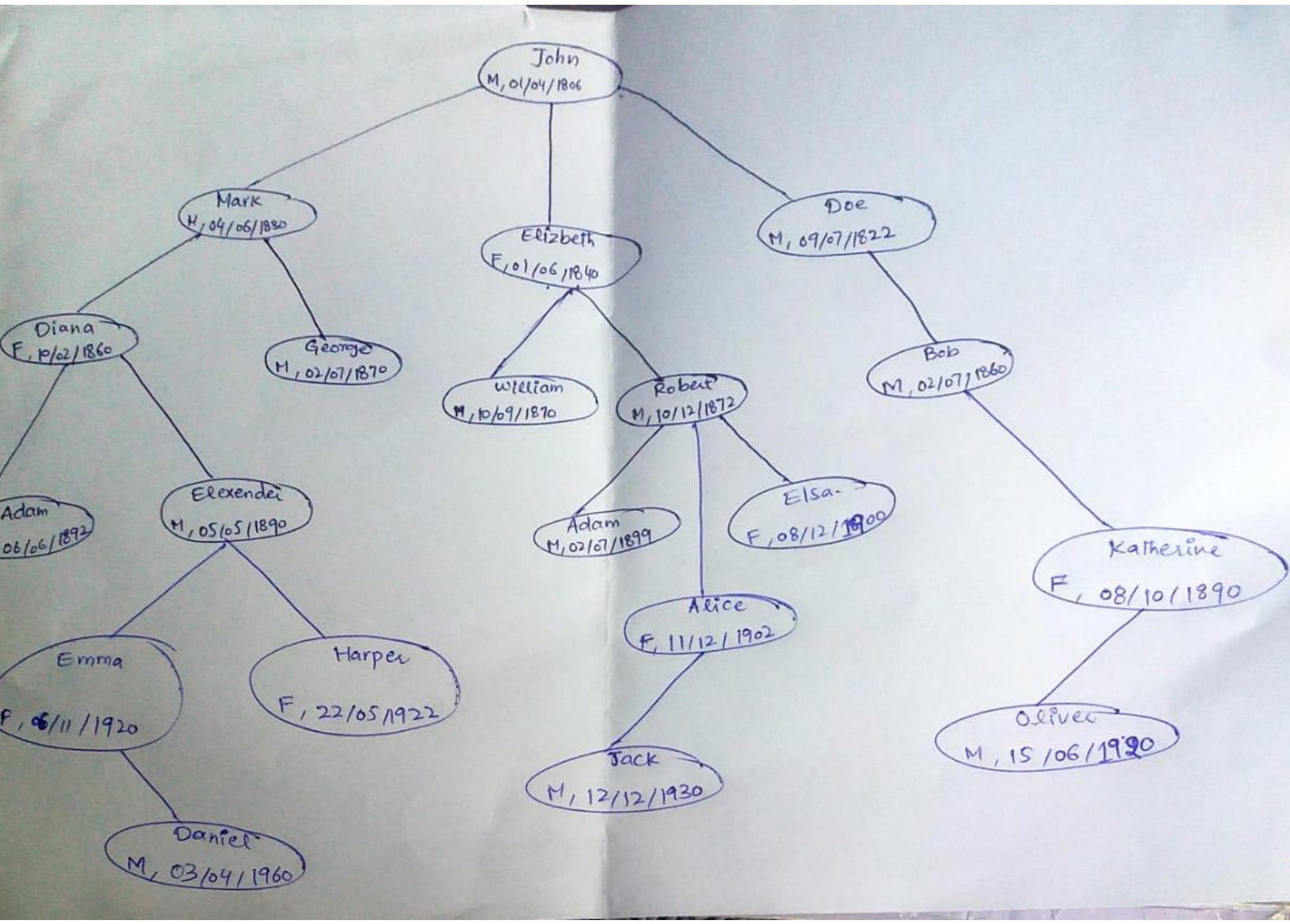
void displayFamilyOf(Person p) // this should only display information about all children of a person p

Person* FindEldestGrandsonOf(Person *grandfather) // the function should traverse a tree 2 levels down the grandfather to find grandsons. The eldest among all is the one with greatest age. For example, the eldest grandson of Mark is Elexender as shown in a sample tree below.

void displayTree(Person *p) // the function should traverse the whole tree **recursively** starting from where person p exists down till the last level. On each level it should first display the father name followed by children names. No loops are allowed in this function

~ FamilyTree **();** // delete dynamic element

};



Total Marks: 45

Q6: Implementation of Matrix Class – Your goal is to implement a generic “Matrix" class. You will need to write three files (matrix.h, matrix.cpp and Q7.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
class Matrix{
   // think about the private data members... // the
   matrix should store real numbers public:
      //include all the necessary checks before performing the operations in the functions
      Matrix();// a default constructor
      Matrix(int, int);// a parametrized constructor
      Matrix(const Matrix &);// copy constructor
      void set(int i, int j, float val);//set value at (i,j)
      float get(int i, int j)const;//get value at (i,j)
      Matrix& assign(const Matrix &);//assigns (copies) a Matrix. Returns the same
      Matrix add(const Matrix &);//adds two Matrices and returns the result
      Matrix subtract(const Matrix &);//subtracts two Matrices and returns the result
      Matrix multiply(const Matrix &);//multiplies two Matrices and returns the result
      Matrix multiplyElement(const Matrix &);//Elementwise multiplies two Matrices and returns the
      result
      Matrix add(float );//assigns a constant to every element
      Matrix multiply(float);//multiplies every element with a constant
      void input(); // takes input in every element of matrix
      void display(); // prints every element
      ~Matrix();
};
```

Total Marks: 10

Q7: struct SavingAccount – create a struct SavingAccount with following data members

char* name

float annualInterestRate

double savingBalance

char* accountNum // the account numbers will be from “SA00” to “SA99”. Each account must have a unique account number that has not been assigned to any other customer before.

The following two will be passed as arguments to some global functions mentioned below

1. SavingAccount *savers[100] – an array of 100 SavingAccount pointers.

2. int accountsOpen – an integer to store the current accounts open and can also be used as an index for the customer’s array.

Implement the following functions in global scope:

void OpenCustomerAccount (SavingAccount * savers[], int accountsOpen, char* NameVal, double balance) – a function to create a new account and assign it an account number.

float calculateMonthlyInterest (SavingAccount * saver) - that calculates the monthly interest by multiplying the balance by annualInterestRate divided by 12

void modifyInterestRate(SavingAccount * saver, float newValue)

int SearchCustomer (SavingAccount * savers[], int accountsOpen, char* accountNum) – a function that searches for an account using an account number. If the customer is found it returns the array index otherwise return -1

bool UpdateAccountBalance (SavingAccount * savers[], int accountsOpen, char *accountNumVal, double balanceVal) – a function that updates a customer's balance

Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

Happy Coding!