Implementation of Family Tree – Create a class Person that should store following attributes of a person (70 Marks)

char* 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 an array of pointers that point to each registered child. Assume that a 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 parameterized 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 records 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 you are required to make a menu driven code.

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.

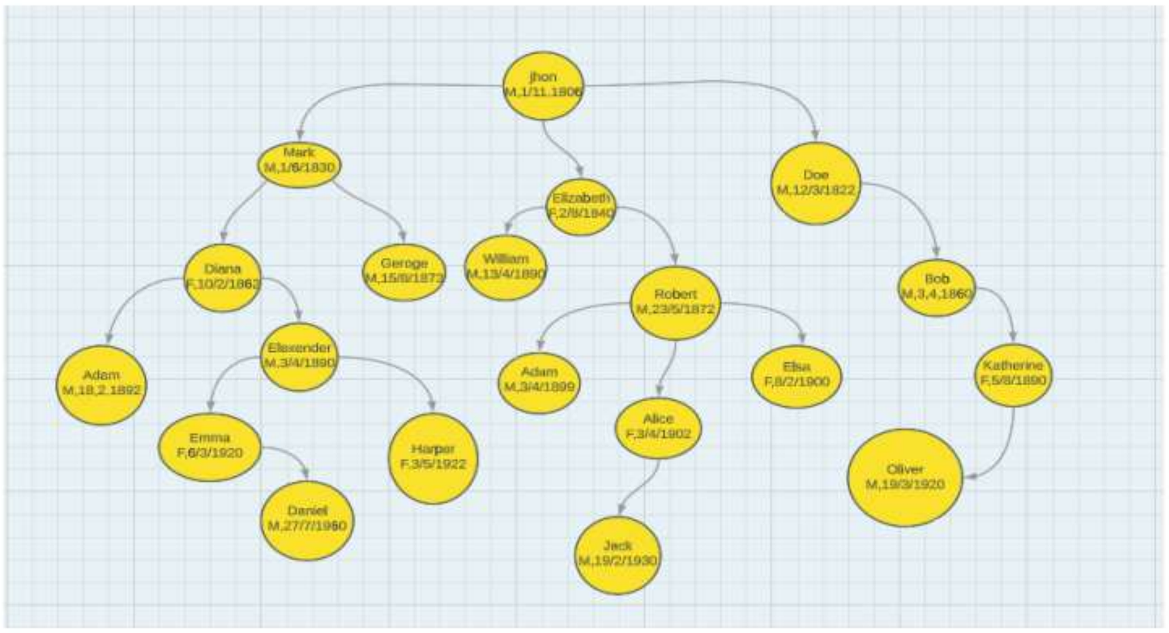
void displayFamilyOf(Person p) // this should only display information about all children of a person p

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

};


jhon M,1/11,1806
Mark M,1/6/1830
Elizabeth F,2/8/1840
Doe M,12/3/1822
Diana F,10/2/1862
Geroge M,15/8/1872
William M,13/4/1890
Robert M,23/5/1872
Bob M,3,4,1860
Adam M,18,2,1892
Elexender M,3/4/1890
Adam M,3/4/1899
Elsa F,8/2/1900
Katherine F,5/8/1890
Emma F,6/3/1920
Harper F,3/5/1922
Alice F,3/4/1902
Oliver M,19/3/1920
Daniel M,27/7/1960
Jack M,19/2/1930