Assignment # 2

Course Title Object Oriented Programming Due Date: 02/05/2022

Instructions:

1. Assignments are to be done individually. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. The code you write must be your own and you must understand each part of coding.
2. The AIM of this assignment is to practice with structures and classes.
3. Plagiarism of any kind (copying from others and copying from the internet, etc.,) is not allowed.
4. Please make separate header and CPP files for each question.

Question 1.

Consider following basic structures to represent a rectangle. You can add more functions or members as per your need.

```
struct Rectangle {
   int x;
   int y;
   int w;// width
   int h; // height
};
```

1. Generate N number of rectangles with different sizes
2. Find the area of all generated rectangles and display them.
3. Sort rectangles based on their area.
4. Implement following method to return true of first rectangle is greater than the 2nd rectangle:
   `bool compareRect(Rectangle r1,Rectangle r2)`
5. Check which rectangles have the same area and display it.
6. Generate a random line. Find the number of rectangles that this line intersect with.

Question 2.

Implement a structure, **Car**. The structure has the following data member:

1. **int petrolLevel** – that indicates the liters of petrol in the car's petrol tank. PetrolLevel forthis car can only be in the range 0 – 45 liters.

The structure has the following member functions:

1. **void setPetrolLevel(int petrolLevelVal)** – a setter for petrol level, cannot set value greaterthan 45.
2. **int getPetrolLevel()** – a getter for petrolLevel
3. **Car()** – a default constructor
4. **Car(int petrolLevelVal)** – a parametrized constructor

5. **bool MoveCar(int distanceKM)** – a function that takes an integer as an argument and moves the car to the argument value which is in Km. Provided the petrol tank has enough fuel. Successful movement of the car returns true otherwise returns false. Moving for each km causes the petrolLevel to go low by one.
6. **void Refill()** – a function that refills the car tank to the maximum value of petrolLevel
7. **bool isEmpty()** – a function that tells whether the Petrol tank is Empty or not

Question 3.

Your goal is to implement a generic "String" class using char array. You will need to write three files (string.h, string.cpp and stringMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive all functions of your class implementation. Please note that we will be running your test code and any segmentation faults or incorrect result will result in loss of marks.

```
class String{
private:
// think about the private data members...
public:
// provide definitions of following functions...
String();          // default constructor
String(char *str);       // initializes the string with constant cstring
String(const String &);                  // copy constructor
String(int x);           // initializes a string of pre-defined size
char getAt(int i);       // returns the character at index [x]
void setAt(int i, char c);       // set the character at index [x]
String substr(int pos, int len);          // returns a substring of length len from 'pos'
String substr(int pos);          // returns substring from the given position to the end.
void append(char a);          // append a char at the end of string
void append(String str );       // append a String at the end of string
void append(char *str );       // append a constant c string at the end of string
int length();             // returns the length of string
char * tocstring();      // converts a String to c-string
void display();           // displays the string ..
bool isEmpty();           // returns true if string is empty..
void copy(const String&);     // Copy one string to another ...
```

void copy(**const char** *); // copy cstring to String...

int find(**char**); // returns the index of character being searched.

bool equal(String); // should return true if both strings are same

int stoi(); // function for converting a string to integer.

˜String(); // destructor...

};

Question 4.

Implement a structure **Address** with the following data members:

1. **char* address** – a sting to store the address
2. **char* city** – a sting to store the city name
3. **char* state** – a sting to store the state name
4. **int zip_code** – an integer to store the ZIP code

Implement a structure **CustomerAccount** with the following data members:

1. **char* name** – a string to store the name of the customer
2. **Address address** – a nested structure variable to hold the address
3. **long long phoneNum** – a long long to store the phone number of the customer
4. **float balance** – a float to store the customer's balance
5. **char* accountNum** – a string to store the customer's account number, account numbersare in the for "PK001" to "PK100".

The following two will be passed as arguments to all the global functions mentioned below.

1. **CustomerAccount *customers[100]** – an array of 100 CustomerAccount 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:

1. **void OpenCustomerAccount (CustomerAccount* customers[], int& accountsOpen, char* NameVal, char*addVal, char*cityVal, char*stateVal, int zipcodeVal, long long phoneVal, float balanceVal)** – a function to create a new customer account and assign itan account number that has not already been taken between PK001 and PK100.
   To dynamically create a new customer account, use an element in the customers array (apointer) with the *new* keyword.

2. **int SearchCustomer (CustomerAccount* customers[], int accountsOpen, char* accountNum)** – a function that searches for a customer using an account number. If the customer is found it returns the array index otherwise return -1
3. **bool UpdateCustomerAccount (CustomerAccount* customers[], int accountsOpen, char *accountNumVal, Address addressVal)** – an overloaded function that updates a customer's address
4. **bool UpdateCustomerAccount (CustomerAccount* customers[], int accountsOpen, char *accountNumVal, long long phoneVal)** – a function that updates a customer's phonenumber
5. **bool UpdateCustomerAccount (CustomerAccount* customers[], int accountsOpen, char *accountNumVal, float balanceVal)** – a function that updates a customer's balance
6. **void DisplayAllCustomers(CustomerAccount* customers[], int accountsOpen)** – a function that displays all customer accounts in the bank

Question 5.

Implement a class **Matrix** to create matrices of 2x2, 3x3 and 4x4 with the following private datamembers:

1. int **matrix – a double pointer of type integer to create a 2D array (matrix)
2. int row – an integer to store the rows in the matrix
3. int col – an integer to store the columns in the matrix

The class shall have the following public member functions:

1. **Matrix (int n1, int n2, int n3, int n4, int row = 2, int col = 2)** – a parametrizedconstructor for a 2x2 matrix
2. **Matrix (int n1, int n2, int n3, int n4, int n5, int n6, int n7, int n8, int n9, int row = 3,int col = 3)** – a parametrized constructor for a 3x3 matrix
3. **Matrix (int n1, int n2, int n3, int n4, int n5, int n6, int n7, int n8, int n9, int n10, intn11, int n12, int n13, int n14, int n15, int n16, int row = 4, int col = 4)** – a parametrized constructor for a 4x4 matrix
4. **Matrix(const Matrix &m)** – a copy constructor
5. **~Matrix()** – a destructor
6. **int getRow()** – a getter for rows
7. **int getCol()** – a getter for columns
8. **int getValue(int row, int col)** – a function to get the value at the given row andcolumn of a matrix
9. **void setValue(int row, int col, int value)** – a function to set the value at a given rowand column index of a matrix.
10. **int Total()** - Calculate the total/sum of all the values in the matrix.
11. **double Average()** - Calculates average of all the values in the array.

12. **int RowTotal(int row)** - Calculates total/sum of the values in the specified row.
13. **int ColumnTotal(int col)** - Calculates total/sum of the values in the specified column.
14. **int HighestInRow(int row)** - Finds highest value in the specified row of the array.
15. **int LowestInRow(int row)** - Finds lowest value in the specified row of the array.
16. **Matrix Transpose()** - Find Transpose of array.
17. **int LeftDiagonalTotal()** - Calculates total/sum of the values in the left Diagonal ofarray.
18. **int RightDiagonalTotal()** - Calculates total/sum of the values in the right Diagonal ofarray.
19. **Matrix Add(Matrix m**) – adds the calling object matrix with the one passed as anargument and returns a resultant matrix
20. **Matrix Subtract(Matrix m**) – subtracts the calling object matrix with the one passedas an argument and returns a resultant matrix
21. **Matrix Multiply(Matrix m**) – multiplies the calling object matrix with the one passedas an argument and returns a resultant matrix
22. **int FindkSmallest(int k)** – a function that finds the k^{th} smallest element in the matrix.If k is 1, return the smallest element. If k is 2 return the second smallest element andso on.
23. **int FindkLargest(int k)** – a function that finds the k^{th} largest element in the matrix. Ifk is 1, return the largest element. If k is 2 return the second largest element and so on.
24. **Matrix merge(Matrix m)** - a function to merge two matrices. E.g.m = $\begin{Bmatrix} 1, 2, 3, 4 \\ 9, 8, 7, 6 \end{Bmatrix}$ and n = $\begin{Bmatrix} 5, 6, 7, 8 \\ 0, 0, 0, 0 \end{Bmatrix}$

after merge = $\begin{Bmatrix} 1, 2, 3, 4, 5, 6, 7, 8 \\ 9, 8, 7, 6, 0, 0, 0, 0 \end{Bmatrix}$

QUESTION 6

In this program, you will develop a calendar application that can display the full calendar for any month of any year and allow the user to calculate the difference between two dates. Implement the following functionality:

1. Display calendar for any month of any year
2. Calculate the difference in months, weeks or days between any two dates
3. Calculate the date and day of the week n months, n weeks, or n days from a given date.

When the program is run, the calendar of the current month (taken from the system) should be displayed by default. Then the user should be given the following options:

1. **Display calendar for a different month**
   Take user input for the month and year whose calendar he would like to view and display it.

2. **Calculate difference between two dates**
   Take user input for both dates and calculate the difference between them in months and days as well as weeks and days. For example, if the user enters x and y, display the difference as:

   39 months, 3 days, OR 78 weeks, 3 days.

3. **Calculate a future date**
   Take user input for the start date and for the number of weeks, months or days to add to it. The user can type "3 months", "43 weeks", "237 days" etc. and your program should interpret the text input correctly and add the appropriate number of days to the start date. Display the output in the form:
   <user input> from the start date <user input> is <day of the week>, <day> <month> <year>.
   For example:
   43 weeks from the start date 3 March 2021 is Thursday, 6 January 2022.

4. **Calculate a past date**
   This is the reverse of part 4 above. Take user input for the start date and subtract the number of days, weeks or months that the user specifies.

Note: you must use overloading of the subtraction operator to implement the data difference functionality (points 2, 3 and 4 above).

Implement the above as member functions in at least a **Date** class and a **Calendar** class. You can use any number and type of data members and also implement additional classes if needed.

Write a driver program with a menu-driven interface that interacts with the user and gives options to test all the above functions; the program should keep displaying the options until the user chooses to exit.