1. Write a recursive function that will generate and print the first n Fibonacci numbers

2. Using pointers, write a function that receives a character string and a character as argument and deletes all occurrences of this character in the string. The function should return the corrected string with no holes.

3. Given an array of sorted list of integer numbers, write a function to search for a particular item using the method of binary search. And also show how this function may be used in a program. Use pointers and pointer arithmetic

4. Write a program to extract a portion of a character string and print the extracted string. Assume that m characters are extracted, starting with nth character.

5. Write a program to print the Pascal triangle for 10 rows

6. Write a Program that will read a positive integer and print its binary equivalent

7. Develop a program to add and multiply two matrices.

8. Develop a program to sort n strings. Use array of pointers.

9. Develop a program to read lines and print only those containing a certain word.

10. Write a program to sort all the elements of 4 by 4 matrix. A magic square is a square array of positive integers such that the sum of each row, column, and diagonal is the same constant.

For Example:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Given above is a magic square whose constant is 34. Write a program to determine whether or not the given square is a magic square.

11. Write a function to get the transpose of a matrix.

12. Write a program to reverse a string using pointers.

13. Develop a program to calculate nPr and nCr given n and r.

14. Write a program that will open a file and report on the number of lines, characters, and words in a file. Have the name of the file to be opened appear as a command line argument.

15. Write a program to swap contents of two variables without using third variable.

16. Write a program, which will delete only odd numbers from a linked list of integers.

17. Write a program to remove trailing blanks and tabs from each line of input, and to delete entirely blank lines.

18. Write a function smallest that takes three integers, x, y, z and returns an integer smallest among them.

19. An integer is said to be prime if it is divisible only by 1 and itself. For example 2, 3, 5 and 7 are prime, but 4, 6, 8, and 9 are not.

20. Write a function that determines if a number is prime.

21. Use this function in a program that determines and prints all the prime numbers between 1 and 100.

22. Write the versions of the library functions strncpy, strncat, and strncmp, which operate on at most the first n characters of their argument strings. For example, strncpy(s,t,n) copies at most n characters of t to s.

23. Write a program that reads three nonzero float values representing the sides of a triangle. Calculate the area of the triangle.

24. Write a program to encrypt / decrypt a file using bit wise operator XOR (^).

25. Create a database management system for managing student's information. Add the features like, add, delete, modify, search, sort etc. The program should also have the feature for generating reports in the text file format. The text file should contain titles under which all the information is listed in a tabular form.

26. Write a program to remove all the comments from a 'C' program.

27. Develop a program that reads a file containing a list of numbers, and then writes two files, one with all numbers divisible by three and another containing all the other numbers

28. Take one of your previous programs and run it using the interactive debugger to examine several intermediate values.

29. Develop a program that counts the number of bits in a character array. Optimize it through the use of register integer variables. Time it o several different arrays of different sizes. How much time does you save?

30. Try examples by embedding assembly code in C and observe the performance

Data Structures:

1. Create a linked list ADT with functions for creation, insertion, deletion & searching
2. Write a recursive function for printing the elements of a linked list in the reverse order.
3. Write a function to reverse the linked list.
4. Write a program to add two polynomials. Use linked list representation for polynomials.
5. Given two sorted lists, L1 and L2, write a procedure to compute L1 L2 using only the basic list operations.
6. Develop an array implementation of self-adjusting lists. A self-adjusting list is like a regular list, except that all insertions are performed at the front, and when an element is accessed by the Find, it is moved to the front of the list without changing the relative order of the other items.
7. Implement a doubly linked list ADT
8. Create a Stack ADT using array implementation and solve the following
    1. Balancing Symbols
    2. Evaluation of postfix expression

3. Converting Infix to postfix

9. Implement Queue ADT using linked list implementation

10. Implement Deque ADT. Deque is a data structure in which insertions and deletions are performed only at the ends.

11. Implement C program to sort N number of student records using

    1. Bubble sort
    2. Insertion Sort
    3. Heap Sort
    4. Quick sort
    5. Merge Sort

12. Search for a particular student details from the sorted student list using Bubble Sort

13. Given a postfix expression, construct an expression tree and include the following functions:

    1. Inorder traversal
    2. Height of the tree
    3. Mirror Image
    4. No of nodes
    5. Preorder traversal
    6. Postorder traversal

14. Create Binary Search Tree ADT with the following functions

    1. Insertion
    2. Deletion
    3. Find
    4. Find_Min
    5. Find_Max

15. Given Inorder and Preorder traversals of a binary tree, construct the binary tree.

16. Write a non recursive routine to traverse the binary tree in inorder

17. Create an AVL tree by including functionality for inserting, singlerotationleft, singlerotationright, doublerotationleft & doublerotationright

18. Develop a program to find the shortest path in a graph

19. Implement the linux file system.

20. Create a hash table of size 11 which maintains keys of integer type. Resolve the collisions using separate chaining. Also include functionality to print the elements hashed into each of the slot in hash table

C++:

1. Create a class called **Time** that has separate int data members for hours, minute, and seconds. One constructor should initialize the data to 0, and another should initialize it to fixed values. A member function should display it in 11:43:34 format. The final member function should add two objects of type **Time** passed as arguments. A main() program should create two initialized **Time** object, and one that isn't initialized. Then it should add the two initialized values together, leaving the result in the third time variable. Finally it should display the value of this third variable.

2. Create two classes **Distance1** & **Distance2** that store the value of distances. Distance1 stores distance in meters and centimeters and Distance2 stores the distance in feets and inches. Write a program that can read values for the class objects and add one object of Distance1 with another object of Distance2.Use a friend function to carry out the addition operation. The object that stores the results may be a Distance1 or Distance2 object, depending on the unit in which the result is required.

3. A bookshop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, publisher, price, and stock position. Whenever a customer wants a book, the sales person inputs the title & author name and system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book detail, and requested no. of copies required. If the requested copies are available, the total cost of requested copies is displayed, otherwise appropriate message is displayed.

4. Write a header file PERSON.h containing definition of independent class Person, and another header file ADDRESS.h containing definition of independent class Address. Write a container class Employee that has a Person and an Address object as contained data members (besides other data members) by including the header files. Demonstrate the passing of arguments in the constructors of Person and Address classes by the constructor of the Employee class.

5. Write a program to demonstrate the situation where it is necessary to write the code for copy constructor.

6. Define a class named Employee with data members empno, name, & salary and appropriate member functions for setting and reading the data, constructors and destructor. Create an array of 5 Employee objects, populate it with data by overloading >> operator and then write all 5 employees data to a file using a single write function call. Also write the code to read the contents of the file using a single read function call, and then display the data of 5 employees individually on the console by overloading << operator.

7. Write a class Polar that describes a point in the plane using polar coordinates radius and angle. Use the overloaded + operator to add two objects of Polar class. Also use conversion function to convert polar to rectangle and vice-versa.*Note: To add polar values you need to convert it into rectangular coordinates.

8. Write an abstract class named shape with one pure virtual function void area(). Create 3 derived classes named rectangle, circle and triangle that derive from the shape class and override the area() function to have their specific implementations. Demonstrate that an object of the base class cannot be created, and if a deriving class does not implement the pure virtual function then that class also becomes an abstract class. Create a pointer of the shape class and demonstrate the polymorphic nature of such classes.

9. Write a program to demonstrate hybrid inheritance with virtual classes using the following classes,

class person: base class

data members: name, age

class physique: derived virtually from person

data members: height, weight

class family: derived virtually from person:

data members: numChildren, religion

class employee derived from physique & family

data members: empno, salary

Create objects of employee class to show the order of invocation of constructors and destructors. Display the sizes of each of these classes and explain with comments.

10. Write 3 separate programs to demonstrate the different types of derivations, public, private and protected. Explain using appropriate comments the access of inherited members.

11. Write a program to demonstrate the use of virtual destructors.

12. Write a program to demonstrate the functions get(), putback(), peek(), and gcount(). Also demonstrate the use of formatting flags and formatting functions of ios class.

13. Write a generic function template to add two values and return their sum with the following,

(a) Demonstrate its implicit specialization with int, double and float data types.

(b) Use default arguments in the template function.

(c) Write an explicit specialization of the generic template for int data type.

(d) Demonstrate that an explicit version of the function is called with int data type.

(e) Overload the generic template.

14. Create user-defined exception classes QEmpty and QFull having a data member msg. Write a Queue class that throws these exceptions during underflow and overflow of the queue, and displays a message describing the type of exception

15. Write a program with the following,

(a) A function to read two double data type numbers from the keyboard.

(b) A function to calculate the division of these two numbers.

(c) A try block to throw an exception when a wrong data type is keyed in.

(d) A try block to detect & throw an exception if the condition "Divide by zero" occurs.

(e) Appropriate catch block to handle the exception thrown.

(f) Also demonstrate the concept of rethrowing an exception.

(g) Use catch (...) handler.

16. Create a **Money** class consisting of an unsigned long and a short data member, **dollars** and **cents**. You may assume that all money values are non-negative. Add the following member functions:

    1. A constructor with two default arguments, an unsigned long and an unsigned short. The arguments should initialize the dollars and cents members. Since both arguments have default values (both 0), this constructor also serves as a default constructor.

2. A constructor with a double argument. The double must be used to initialize both the dollars and cents. For example, an argument of 1.75 should assign 1 to the dollars and 75 to the cents. Round off cents to the second decimal place, so 5.55555 would set the dollars to 5 and the cents to 56.

3. A copy constructor.

4. An overloaded !(unary) operator that serves as a print function. It should print the dollars and cents with a leading dollar sign and a decimal point separating the dollars and cents. Make sure you are able to print the following values: $1.03, $10.00, $0.01, $1234.56, and $0.00.

5. An overloaded + (unary) operator that "reduces" Money. For example, if you passed 5 and 150 into the first constructor, you would want to use this function to change the Money object to have dollars = 6 and cents = 50. This function should return Money by reference.

6. An overloaded < (binary) operator that tests two Money objects to see if the first is less than the second. This function should return a bool.

7. An overloaded == (binary) operator that tests two Money objects to see if the first is equal to the second. This function should return a bool.

8. An overloaded + (binary) operator that adds two Money objects and returns the sum by value (Money). Make sure your logic can handle $2.56+$5.67 and $5.63+$0.37.

9. An overloaded - (binary) operator that subtracts two Money objects and returns the difference. I should return Money by value. If you try to subtract larger Money from a smaller one, print an error message and have the function return $0.00. (0 dollars and 0 cents).

10. An overloaded * (binary) operator with a double argument. This permits you to multiple a Money object by a double, For example, $4.29*67.3. It should return Money by value.

11. An overloaded += operator that adds more Money to a Money object and returns the result by reference. For example, M1 += M2; (this should change M1).

Divide your final program into three files: a header file for your Money class, a source file for your Money methods, and another source file for main(). Write you own main() and thoroughly test all functions. Make sure your main() demonstrates calls to each member function.

17. This assignment will give you practice working with classes, constructors, static data member, static member functions, and friend functions. The goal of this assignment is to create a partial model of a population system, emulating the aging and dying of a population, but unfortunately not the birth process. But, may be that's fortunate for the programmer

Program requirements:

Create the three classes described below.

Use the main() function included.

Your output should look like that shown below.

Your program must be divided into multiple files. Each class should be defined in a separate header file and the member functions for each class should be in a separate source file. main() should also be in a separate file. Your program should consist of at least 7 files. Print each file on a separate page.

Class Descriptions

The date class is used to represent calendar dates. As a minimum, the class must contain:

1. 3 unsigned int data members to represent month, day, and year
2. a default constructor (use the same one from assignment 3)
3. a constructor that takes 3 unsigned ints as arguments
4. an increment function that adds 1 day to the date (you can use the same one from assignment 3). For determining ages, you can assume that a year is 365.25 days long.
5. a display() function that displays the date in the mm/dd/yy format
6. a let_time_pass() function that adds a random number of days to a date. The random number should be between 1 and 365. You will be using this function to add days to the (global) TODAY date. Hint: you might use the random number to call the increment() function repeatedly.

7. Declare the human class as a friend of the date class.
8. Declare the function, int difference_between_2_dates(date,date) as a friend of the date class.

The human class must contain at least:

9. 3 data members:

char name[32]

date birthday

bool alive

10. 2 static data members

static human* oldest_human

static unsigned long number_of_living_humans

11. a constructor: human(const char* n,const date& b)
12. necessary accessor functions
13. a function, age() that returns a human's age in years
14. a die() function (you know what that means)
15. a display() function
16. a static member function that assigns the appropriate human* to the oldest_human.
17. a static member function that returns the number_of_living_humans
18. Declare the ffunction void population::display() const as a friend of the human class.

The population class must contain:

19. Two data members:

human** people

const unsigned long original_size

20. A private member function: determine_oldest() that “sets” the oldest_human
21. A constructor
22. A destructor
23. A display() function
24. An examine_population() function that takes a look at the population, calls the following roll_the_dice() function for each “living” human. If roll_the_dice() returns a number greater than .5, the human should “die”.

```
float roll_the_dice(unsigned short age)

{

return (float) age*(rand()%100)/10000.;

}
```

Assumptions

- All humans were "born" in the last century.
- Use the difference_between_2_dates() function so that it always returns a positive value. That is, you should always subtract the older date from the newer date.
- Use a population size of 20 for the final testing of your program.

Hints and Suggestions

If you are new to working with multiple files, keep your program as one file until you get it working, then try to split it into multiple files.

Declare TODAY as a global variable. To do this, enter:

```
date TODAY;
```

in your main() source file and declare it as an extern in the date header file, like this:

```
extern date TODAY;
```

Create a global array of names that you can use in the population constructor, like this:

```
char* NAMES[] = {"Fred","Sam","Sally","George","Sue","Mary","Bill",...};
```

Do not work with a population size of 20 until you are sure that your program is working. Use a small population, like 4 or 5.

The main() function

```
int main()
{
srand(time(0)); // seed the random number generator
population World(POPULATION_SIZE);
cout << "Today is ";
TODAY.display();
cout << endl;
World.display();
// let time pass until half of the world's population dies
do
{
TODAY.let_time_pass();
World.examine_population(); // record deaths, find oldest
} while (human::get_number_of_living_humans() > POPULATION_SIZE/2) ;
cout << "Today is ";
TODAY.display();
cout << endl;
World.display();
return 0;
}
```

Sample Output

```
Today is 11/22/03
Allen was born on 9/15/04 is 99
Catherine was born on 11/1/62 is 41
Naihui was born on 10/16/56 is 47
Gayatri was born on 9/26/24 is 79
Bin was born on 5/12/28 is 75
Evan was born on 5/2/82 is 21
Sandy was born on 3/8/50 is 53
```

Sridevi was born on 4/26/85 is 18

Tanya was born on 3/26/32 is 71

Jing was born on 9/18/25 is 78

Haiying was born on 7/24/75 is 28

Rose was born on 1/21/25 is 78

Nisha was born on 5/8/72 is 31

Hnin was born on 3/14/68 is 35

Adeline was born on 8/17/57 is 46

Chen Wei was born on 8/8/19 is 84

Joe was born on 7/6/86 is 17

Bob was born on 11/26/33 is 69

Mary was born on 7/8/41 is 62

Sue was born on 4/7/80 is 23

The oldest living person, Allen is 99 years old.

4/9/04 Allen died at the age of 99

4/9/04 Chen Wei died at the age of 84

4/9/04 Bob died at the age of 70

9/16/04 Sandy died at the age of 54

9/16/04 Tanya died at the age of 72

9/16/04 Rose died at the age of 79

9/16/04 Mary died at the age of 63

1/10/05 Bin died at the age of 76

8/5/05 Gayatri died at the age of 80

8/5/05 Jing died at the age of 79

Today is 8/5/05

Catherine was born on 11/1/62 is 42

Naihui was born on 10/16/56 is 48

Evan was born on 5/2/82 is 23

Sridevi was born on 4/26/85 is 20

Haiying was born on 7/24/75 is 30

Nisha was born on 5/8/72 is 33

Hnin was born on 3/14/68 is 37

Adeline was born on 8/17/57 is 47

Joe was born on 7/6/86 is 19

Sue was born on 4/7/80 is 25

The oldest living person, Catherine is 42 years old.

Text Books:

1. “C++ Complete Reference”, Herbert Schildt, Tata McGraw-Hill

2. "Data Structures and Algorithm Analysis: in C++, Third Edition”, Mark Allen Weiss, Addison-Wesley

Reference Books:

1. “The C++ Programming Language”, Bjarne Stroustroup, Addison-Wesley

2. Data structures & Algorithms by R.S. salaria

3. C Programming Language, Schaum Series

4. C Programming Language by Kernighan and Ritchie

5. Intel manual for X86 architecture