Assignments – Day 1

[TOPIC - String Operations]

1) Write a function to check if 2 strings are identical or not. The strings to be checked should be passed as command line arguments to the program.

   [**NOTE:** Your program should NOT use any string compare library functions]

2) Write a function perform the following operation -

   input string – hello
   output string - ifmmp

   Here the output should print the next alphabet in the alphabetical system. For ex – 'i' comes after 'h', 'f' comes after 'e' in the alphabetical system. Input string is passes as a command line parameter to the program.

   [**NOTE:** Your program should NOT use any string compare library functions]

3) Write a function to concatenate 2 strings in the following format –

   If the two strings are

   String 1 is - hello

   String 2 is – world

   Output should be –

   ollehworld

   The 2 strings to be concatenated are passed as command line arguments to program.

   [**NOTE:** Your program should NOT use any string compare library functions]

4) Write a function to check if a given string is a palindrome or not. The string to be checked should be passed as command line arguments to the program.

   [**NOTE:** Your program should NOT use any string compare library functions]

5) Implement the following function -

   char *my_strstr(const char *s1, const

   char *s2) Returns a pointer to the first

   instance of string s2 in s1 Returns a NULL

   pointer if s2 is not encountered in s1

   [**NOTE:** Your program should NOT use any string compare library functions]

6) Write a program to prompt the user to input a string and find the number of occurrences of alphabetic characters in the string. The letters occurring should be displayed in alphabetic order.

NOTE 1: Alphabets that do not appear in the input string are not mentioned in the output NOTE 2: Output is displayed in alphabetic order

NOTE 3: Alphabetic characters occurrence is case-insensitive (i.e. both ‘e’ and ‘E’ counts as ‘E’) NOTE 4: Non-alphabetic characters (such as blank space in above example) are ignored.

Assignments – Day 2

[TOPIC –Bitwise operations and Logic building]

1) Write a program as per below requirements.

   The program should ask the user to enter a number.
   Once the user enters a number, the program should display the below menu –

```
            MAIN MENU
    1. Toggle a bit
    2. Set a bit
    3. Clear a bit
    4. Identify the bit
    5. Exit
    Enter your choice (1..5) : __
```

   If the user enters 1, the program should ask the user – “Which bit do you want to toggle? : __”

   If the user enters 4, the program should toggle bit 4 in the initially inputted number (i.e. 4th bit starting from Least Significant Bit), and should display the updated number

   If the user enters 2, the program should ask the user – “Which bit do you want to Set? : __”

   If the user enters 4, the program should Set bit 4 in the initially inputted number (i.e. 4th bit starting from Least Significant Bit), and should display the updated number

   If the user enters 3, the program should ask the user – “Which bit do you want to Clear? : __”

   If the user enters 4, the program should Clear bit 4 in the initially inputted number (i.e. 4th bit starting from Least Significant Bit), and should display the updated number

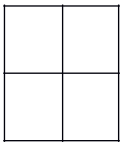
   If the user enters 4, the program should ask the user – “The bit at which position do you want to identify? : __”

   If the user enters 4, the program should Identify the bit at position 4 in the initially inputted number (i.e. 4th bit starting from Least Significant Bit), and should display the bit (0 or 1)

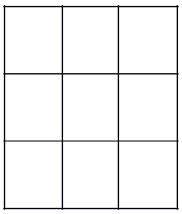
   If the user enters 5, the program should Exit. The menu should repeatedly keep reappearing after all other menu options.

2) Write a program to calculate the total number of all possible squares in a square matrix. [**NOTE:** A square matrix is a matrix with the same number of rows and columns, i.e. n x n matrix]

For example, the below 2x2 matrix table contains 5 possible squares –



For example, the below 3x3 matrix table contains 14 possible squares –



The program must prompt the user to input the order (n) of the matrix and should be able to calculate the total number of all possible squares in the square matrix. The order can be any positive integer value, say - 10x10 or 24x24 or 50x50 etc.

[TOPIC – Pre-processor directives and Macros]

1) Create a debug print which can be used in your program only during debugging. The messages from the debug print should appear only when the **DEBUG** macro is enabled.

   debug_print(“ “);

   The debug print’s output should be the function name and line number in the following format -

   DEBUG PRINT – main()

   and line no. 10 DEBUG

   PRINT – add() and line

   no. 45

   These prints should appear only when the macro DEBUG is enabled.

2) Using preprocessor directive – (conditional compilation) – come up with a mechanism to make sure that a header file (*.h file) included in multiple *.c or *.h files of a project does not result in double inclusion. NOTE: This is also known as **#include guard**, sometimes called a **macro guard**.

3) Write macros to perform the following operations –

   a) Set a specific bit of a number. The number and bit position to be set are passes as input to the macro.

   b) Clear a specific bit of a number. The number and bit position to be cleared are passes as input to the macro.

   c) Toggle a specific bit of a number. The number and bit position to be toggled are passes as input to the macro.

   d) Check a specific bit of a number. The macro should print the value (1 or 0) present at a specific bit position in a number.

4) Write simple programs to illustrate the difference with

   respect to usage of – if()….else…

   and

   #if

   #else

   #endif

   Also explain the advantage of using a preprocessor directive over if().. else…

5) Using bit wise operators implement macro's which performs divide by 2 and multiply by 2 operations on a given number. The macro's should accept the number as a parameter (as given below)

   DIVIDE_BY_TW

   O(x)

   MULTIPLY_BY_

   TWO(x)

6) Write simple programs to illustrate the difference between static and auto variables.

Assignments – Day 3

[TOPIC – C++ File handling and Char/String handling, Function]

1. **Write a program that collects following statistics about given C++ source files.**
   - Total lines
   - Total number of commented lines – you can exclude C style of commenting.
   - Total number of blank lines
   - Total number of non-comment/non-blank lines
   - Total number of lines that has ";"
   - Total number of lines that has just "{" or "}" and white space in it.

   The program should take one or more file as arguments.

2. **Write a program to display names of all the functions** defined in a given C++ code file. Your program should not report function names present in comments and string literals.

   Example:

```
Fn()                                    ->Fn() Should be displayed

{     // Begin function Fn()            -> Should not be reported.
cout<<"In Fn()\n";              -> Should not be reported.
}
main()
{
   Fn();                                -> Should not be reported.
}
```

3. **Write a program that reads a file and creates a new file with the same data, except reverse the case on the second file, print the next character of the original character during writing into the new file and remove the extra spaces between the words. Everywhere uppercase letters appear in the first file, write lowercase letters to the new file, and everywhere lowercase letters appear in the first file, write uppercase letters to the new file.**
   For example,

   file1 (old/first file) contains.

   A To z CaT.

   Then **file2** (new file) should contain.

   b uP A dBU. New file

4. **Raising a number *n* to a power *p* is the same as multiplying *n* by itself *p* times. Write a function called** `power()` **that takes a** `double` **value for n and an int value for p, and returns the result as double value. Use a default argument of 2 for p, so that if this argument is omitted, the number n will be squared. Write a** `main()` **function that gets values from the user.**

5. **Write a program to swap 2 values using** *call by reference* **and the function should be** *inline* **function. Function called** `ref_swap()` **should be used as function name for swap and** `my_overload()` **should be used for function over load. The overload function should use for int, char, float, long.**

Assignments – Day 4

[TOPIC – C++ – (Data structures) - Linked Lists, Class & Objects, Inheritance, Pointer]

1. Create a class that includes a data member that holds a "**serial number**" for each object created from each class. That is, the first object created will be numbered 1, the second 2 and so on. As each object is created, its constructor can examine this count member variable to determine the appropriate serial number for the new object. Add a member function to report/prints the serial number.

   Write a `main ()` function to create three objects and queries each one about its serial number.

   It should print –

   ```
   I am object number 1
   I am object number 2
   I am object number 3
   ```

   And so on for all the objects created.

2. Write a C++ program to perform **queue** operations, create a class called **queue** and two functions called **get()** and **put()** should be used for data insert and delete. You should use dynamic memory allocation to perform the queue operation.

3. **Write a C++ program to implement a Double-Linked List (DLL)**

The program should display the below menu and should support these double-linked-list operations.

```
MENU - Double Linked List
1. Add Node
2. Delete Node
3. Display Node
4. Search Data
5. Exit
   Enter your option (1...5): __
```

Note – The same program must be rewritten using STL also.

4. **Inserting to linked list in reverse sorted order using C++:**

   There are two sorted linked lists (singly linked lists) and their head pointers are represented as p_head_list1 and p_head_list2. Each node of the list contains two elements namely data (integer) and p_next (pointer to next node). Both lists are sorted in descending order based on value of "data".

Write a function that takes two head pointers of the list and merges them into single list and returns the head pointer of the merged list. The merged list should be in the sorted order.

The program should take list1 and list2 as inputs from a file. Your program should take the name of the input file as a command line argument. The file would contain 2 lines of white-space separated list of integers. Integers in line1 should be inserted in list1 and integers in line2 should be inserted in list2. The integers in the file need not be sorted. Your program should take care of inserting the integers into both the lists in reverse sorted order.

Example:If the input file is input.txt, and following are its contents: 19 1 3 4 18 93 12 18 9 6

List 1 should contain: 19 → 18 → 4 → 3 → 1
List 2 should contain: 93 → 18 → 12 → 9 → 6

The merged list should contain these elements in the following order: 93 → 19 → 18 → 18 → 12 → 9 → 6 → 4 → 3 → 1

OUTPUT of your program:

Your program should display list1, list2 and list3 in the following format: list-1: val1 val2 val3 val4

list-2: val1 val2 val3

list-3: val1 val2 val3 val4 val5 val6 val7

NOTE:

1) Your program should strictly stick to the above format of output. It should NOT print anything else apart from this.
2) The same program must be rewritten using STL also.

5. Write a C++ program to reverse the contents in a linked list –

Implement a reverse_list() function that reverses a list by rearranging all the next pointers and the head pointer. reverse_list() should only need to make one pass of the list.

The program should take the list as input from a file. The name of the input file should be taken as a command line argument. This file would contain a line of white-space separated list of numbers. Your program should read the numbers into a list with head pointed by list_head.

It should contain the following interfaces:

```
node*
create_list(FILE *fp)
```

where fp is the file pointer of the input file Function should return the head of the list

```
display_nodes(node *list_head)
```

Displays all the nodes pointed to by list_head

```
int reverse_list(node **list_head)
```

On successful reversing of the list, list_head points to the reversed list

Function returns (-1) on error

Example:

If the input file is input.txt, and following are its contents: 19 1 3 4 18

After reading input from the file, call to display_nodes(list_head) should display: 19 → 1 → 3 → 4 → 18

Call to reverse_list(&list_head) and then calling display_nodes(list_head) should print the following: 18 → 4 → 3 → 1 → 19

OUTPUT of your program:

Your program should output both the original list and the reversed list in the following format: original: val1 val2 val3 val4

reversed: val4 val3 val2 val1

NOTE:

1) Your program should strictly stick to the above format of output. It should NOT print anything else apart from this.
2) The same program must be rewritten using STL also.

6. **Sorted and Unique Linked List:** Write a C++ program to take input from the user to create a sorted linked list with elements in the list sorted in increasing order. Consider the elements in the list to be integers. **(STL and non STL Version both).**

The program then should delete any duplicate nodes from the list. The list should be traversed only once. Take the input of the original list from the user. The program should output the elements in the list after removing the duplicates.

The program should take the list as input from a file, the name of which is given as a command line argument to your program. The file would contain a line of white-space separated list of integers. The numbers in the file need not be sorted.

Your program should take care of inserting the integers into the list in sorted order. Then, it should have a function remove_duplicates(node *list) to remove any duplicate entries in the list.

Your program should read the numbers into a list with head pointed by list_head. It should contain the following interfaces:

node* create_list(FILE *fp)

where fp is the file pointer of the input file

Function should return the head of the list display_nodes(node *list_head)
Displays all the nodes pointed to by list_head

int remove_duplicates(node**list_head)
list_head will point to the list with no duplicates Function returns (-1) on error

Example:

If the input file is input.txt, and following are its contents: 19 1 3 4 8 3 19

After reading input from the file, call to display_nodes(list_head) should display: 1 3 3 4 8 19 19

Call to remove_duplicates(&list_head) and then calling display_nodes(list_head) should print the following:
1 3 4 8 19

OUTPUT of your program:

Your program should output both the original list and the list after removing duplicates in the following format:

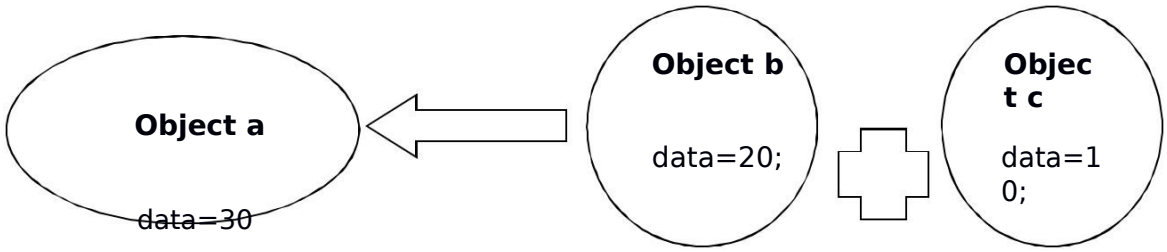
original: val1 val2 val3 val4 val3
val2 val1 processed: val1 val2
val3 val4

NOTE:

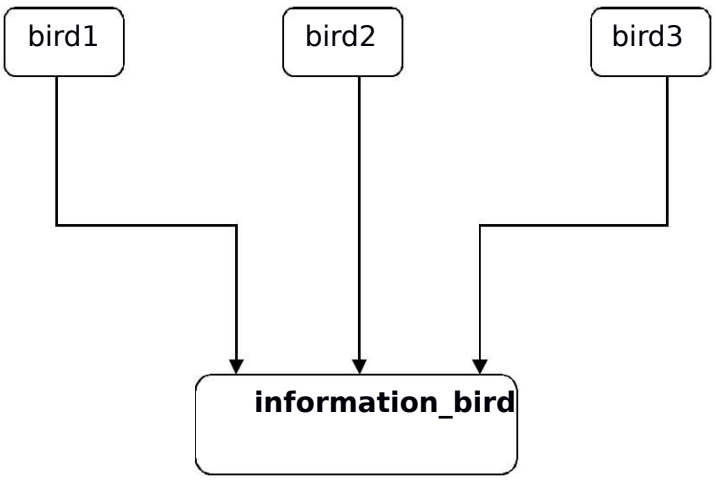
1) Your program should strictly stick to the above format of output. It should NOT print anything else apart from this.
2) The same program must be rewritten using STL also.

7. **Write a program to perform operator overloading -** Program should perform '+' operator overloading. Create a class called `operator` and create 3 object in main function `a, b, c`. Class should contain member function for data input `get_data()`, operator function operator overloading and integer data member for storing the data `int data;`. Expression should be `a=b+c;` and will act as a normal arithmetical expression among the 3 objects. Data of b and c objects will be added and stored in a.

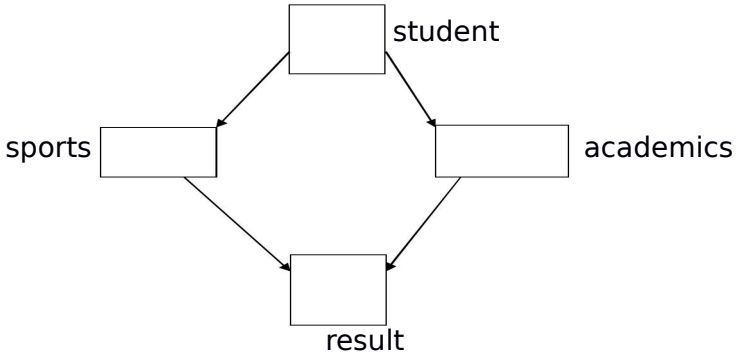
```
class operator
{
      int data;                 //data member
      void get_data();          //member function for get data
                                //operator overload '+' function
};


main()
{
      operator a, b, c;
      a = b + c;

}
```



8. **Write a program to implement multiple inheritances** - Constructor for base classes should be used in derived class only. Create 3 base classes called `bird1`, `bird2 and bird3` and use variables to store information about birds like color, can fly or not, etc in data section and use **`get_info()`** to get the data. Create 1 derived class called `information_bird` which will contain **`put_info()`** will is a function will print the information about the birds.



9. **Write a program to implement hybrid inheritance -** Constructor for base classes should be used in derived class only. Create 1 base class called `student` and use variables to store information about student like name, roll, class, etc in data section and use `getSTD_info()` to get data about student. Create 2 derived class called `sports and academic` which will derived from student and contains `getSPT_info() and getACD_info()` and which will get the data about the student's records about sorts and academic, which will contains sports marks and academic marks . Finally create a derived class called `result` which is derived from class called `sports and academic`. Access the member function and data from result derived class of all base classes and print the data of classes `student`, `sports and academic`.
Print the student's information from class `student`, academic marks/records and academic marks/records from classes' s`ports` and `academic`.



10. Write a program to implement a LINKLIST. Classes and objects should be used for implement the list. It should delete all the links when a linklist object is deleted. It can do this by following along the chain, deleting each link as it goes. You can test the destructor by having a display a message each time. It should delete the same number of links that were added to the list.

```
~del_list()

{

        cout<<"Object Destructed...\n"; //This message will be printed on any
        object get
deleted

}
```

11. Write a program to implement your own version of the library function `strcmp(s1, s2)`, which compares two strings and return 1 if **s1** and **s2 are same but cases are not same**, 0 if **s1** and **s2** are the same, return -1 if **s1 and s2** are not same. Call your function `compstr()` . It should take two `char*` strings as arguments, compare the strings character by character, and return an `int`. Use pointer notation throughout.

```
int compstr(char *s1, char *s2);
          //function definition for string comparison which using 2
   character pointer type arguments string

int main()
{
      compstr("Geeks", "Geeks"); //Function call
}
```

Assignments – Day 5

[TOPIC – C++ – Pointers, Virtual functions, Templates, STL]

1. Write a program to implement the following topics in a single program, 1) Normal member function accessed by pointers from derived classes called `foo()`, 2) Virtual member function accessed by pointers from derived classes called `Vfoo()`, 3) Pure virtual function called `PVfoo()`. Derived class name will class `Derive`.
   Say:

   Normal member function accessed by pointers from derived classes. 2) Virtual member function accessed by pointers from derived classes, 3) Pure virtual function.

```
     class Base
     {
             public:
             void foo()//Base class member function
             {
                     cout<<"Foo Parent";
             }
                     //Need to implement Virtual and Pure Virtual member
function here
        };
```

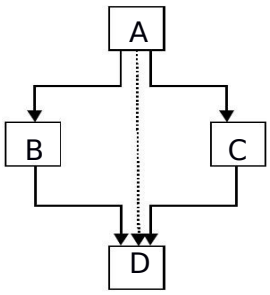
2. Write a program to implement the following topics in a single program, a) Virtual Destructor (Create base and derived class for the implementation), b) Virtual base class (Create base and derived class for the implementation).

   a) Virtual Destructor (Create base and derived class for the implementation)

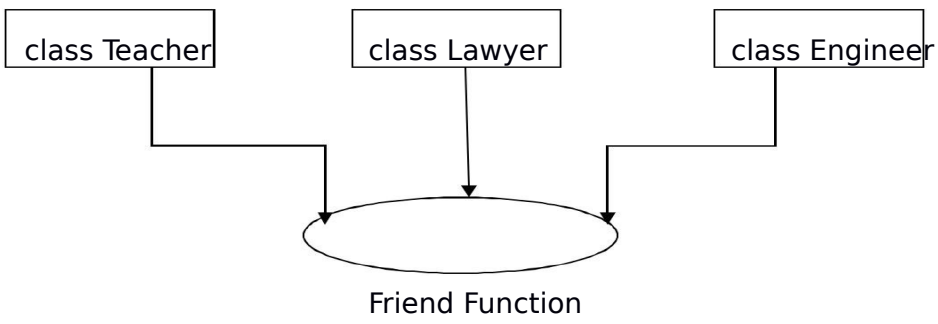
```
class DTBase //Base class name
{

};
class DTDerive //Derived class
```

b) Virtual base class (Create base and derived class for the implementation).
To implement Virtual base class use below mentioned class names and pattern.

```
class ABase //Base Class
{
}
class BDer //Derive class derived from class ABase
{
}
class CDer //Derive class derived from class ABase
{
}
class DDer //Derive class derived from class BDer and CDer
{

}
```



3. Write a program to implement friend function - Create 3 different classes named `Teacher`, `Lawyer` **&** `Engineer`, Friend function name would be `tax_cal()`. Take input from users for `Teacher`, `Lawyer` **&** `Engineer` like Name, Total Income & Calculate the tax in friend function and show the result for individuals.
   Below picture describes 3 classes are accessing 1 friend function to do the same task (tax calculation) of each class.



4. Write a program to implement to implement constructor, destructor, and copy constructor in a single C++ program.

```
class TClass
{
      //write constructor code here
      // write destructor code here
      //write copy constructor here
}
```

Implement main and class body for constructor, destructor and copy constructor.

Output:

If constructor invoked...
Constructor Invoked

If destructor invoked...
Destructor Invoked

If copy constructor invoked...
Copy constructor Invoked

5. Write a template function that returns the average of all of all elements of an array. The arguments to the function should be the array name and the size of the array (type int) and return type would be float.

```
template<typename T, size_t Size>
setNumber();
avg();
```

Output:

Array contents:

1 2 3 4 5 6 7 8 9 10
Average of elements
= 5.5

Note: Use same names for template, array name and function while write the code.

6. Write a program that applies the sort () algorithm to a floating point values entered by the user and display the result. Use array for sorting.

```
int SortArray[8];//Array name
```

```
Output:

Before sort: 32 71 12 45 26 80 53 33 After sort: 12 26 32 33 45 53 71 80
```

Note: Use same names for array while write the code.