Following is the declaration of **MyString** class with its private member variables:

**class MyString {**

**private: char \*str; *// Pointer to the char array that holds the string***

**int length; *// Variable to store the length of the string***

**};**

1. Overload the **bitwise-AND & operator** for **MyString** class, i.e **void operator & (char)**, which should encrypt all letters in the current object of **MyString**. In order to encrypt the letters, this function receives a character parameter and then **adds this character parameter to all the letters in the current object**. Assume that the *copy constructor* of the **MyString** class is already implemented. You are NOT allowed to use any <cstring> library function in your implementation.

Sample Driver

|  |
| --- |
| **int main()**  **{**  **MyString s1("Hello"); // s1 contains Hello**  **MyString s2("PUCIT"); // s2 contains PUCIT**  **s1 & 'b' ; // now “Hello” in s1 is encrypted by ‘b’;**  **s2 & 'm' ; // now “PUCIT” in s2 is encrypted by ‘m’;**  **}** |

|  |
| --- |
| **void MyString::operator & (char c)**  **{  *// complete the function definition here*** |

1. Implement the member function **insertString** of the **MyString** class (see the function prototype below)

**void insertString (int index, char\* newStr);**

This function takes an index and inserts the c-string newStr (second parameter) at the position specified by the index. In this function you will need to re-allocate the array str in the current object and move the characters that occur at or after the specified index.

**Example 1:** If **MyString** object **str1** contains “HelloPakistan” (length = 13) then after this statement:

**str1.insertString (5,"Dear");**

the object **str1** should contain “HelloDearPakistan” (length = 17)

**Example 2:** If **MyString** object **str2** contains “Pakistan” (length = 8) then after this statement:

**str2.insertString (0,"DilDil");**

the object **str2** should contain “DilDilPakistan” (length = 14)

|  |  |  |
| --- | --- | --- |
| **void MyString::insertString (int index, char\* newStr)**  **{  *// complete the function definition here***    **}** | | |
| **QUESTION # 4** | **5+5+5 = 15 Marks** | **Suggested Time: 25 minutes** |

Following is the declaration of **MyList** class:

**class MyList {**

**private:**

**double \*list; *// Pointer to the double array which contains the elements of the list***

**int size; *// Size of the array pointed to by the list pointer***

**public:**

**MyList() { list = 0; }** //Already Implemented

**MyList(int size);** //Already Implemented

**MyList(const MyList &);** //Already Implemented

**MyList operator = (const MyList &);** //You are required to implement

**MyList operator ^ (double exp);** //You are required to implement

//You are also required to implement extraction operator for input.

**};**

Sample driver

|  |
| --- |
| int main()  {  MyList ar0(2), ar1(4); **// ar0 and ar1 will contain 2 and 4 values respectively**  cin >> ar1; **// 4 numbers will be input for ar1, for example:** 1 16 25 49  MyList ar2, ar3; **// ar2 and ar3 will be created**  ar2 = ar1 ^ 0.5; **// ar2 will contain:** 1 4 5 7  ar3 = ar2 ^ 3; **// ar3 will contain:** 1 64 125 343  ar0 = ar1; **// ar0 will contain:** 1 16 25 49  } |

1. Overload the **assignment operator** **=** as a member function of the **MyList** class which assigns the values of right hand side object to the current object. You should first delete any existing values in the current object.

|  |
| --- |
|  |

1. Overload the **extraction operator** **>>** as a member function of the **MyList** class which takes input for the array in the current object of **MyList**.

|  |
| --- |
|  |

1. Overload the **power operator** **^** as a member function of the **MyList** class which has two operands i.e. the current object of **MyList** and an exponent (a double number). This function creates a temporary object with the same number of values as of the current object. It then calculates the power of each number in the object and assigns these values to the temporary object. In the end, it returns the temporary object.