Data Structures & Algorithms

BIT F21 | Lab # 01

June 15 , 2023

Instructions :

- You are not allowed to use C++ STL in this task .
- You can create a private method according to your will **only if** it is genuinely required (Provide a reason for its existence) .
- To get the full credit , you have to keep in mind all the corner cases and implement the counter-checks as well as throw an exception where required .
- **YOU ARE NOT ALLOWED TO USE INTERNET OR MOBILE PHONE . YOUR MARKS WILL GET DEDUCTED IF YOU ARE FOUND GUILTY OF DOING SO !**

Task 1 :

The objective of this task is to revisit the concepts of template classes . You are required to create an ADT named **MySet** just like the one (unordered_set) in C++ STL . MySet should be able to store values of **any** data type provided by the user.

Moreover, by definition, it must contain only unique values. You need to implement the MySet class to support the basic operations described below. It is up to you to decide which container to use to store the set elements and you have to make decision about data members of **MySet** class. Choose the data members carefully !

MySet() ;

MySet(const MySet<T>& obj);

bool insert(const T& val);

This method will return true on successful insertion and if the **val** already exists in the set, it will simply return false without performing insertion.

void erase(const T val);

Simply find and remove **val** from the set (if exists) .

void clear();

Removes all the elements of the set .

void get_size();

Returns the no. of elements / cardinality of the set.

bool contains(const T val);

Returns **true** if val exists in the set , otherwise **false.**

bool is_empty();

Checks wether the set is empty or not.

MySet<T> union(const MySet<T>& other) ;

Returns a set which is result of union of **caller object set** and **other** set**.**

MySet<T>& operator = (const MySet<T>& other) ;

friend ostream& operator << (ostream& out , MySet<T>& obj) ;

~MySet(); // destructor . You know what to do :)

Sample Run

SR # 1 :

```
MySet<T> s ;
s.insert(1);
s.insert(2);
s.insert(2);
s.insert(5);

cout << s ; // should print 1 2 5
s.remove(1);
// now set contains 2 5
```

Task-2

This task is centered around the concept of operator overloading .

Polynomial ADT

You are required to implement the polynomial ADT . An array data structure is to be used for storing the terms in such a way that the coefficient of the term with exponent k will be stored in array at index k.

For example , to store the polynomial 15x^2 + 5x . We'll have to store value 15 (which is co-efficient of 15x^2) at the index 2 and 5 (coefficient of 5x) at the index 1 .

To figure out wether the term with exponent **n** exists in the polynomial or not , one has to simply check wether the value at index **n** is zero or not . If it's zero , it means that coefficient of term with degree **n** is zero which implies that the term does not exist .

Keeping in mind the above information , your job is to provide the implementation of the following methods .

Polynomial (int max_degree) ;

"**max_degree"** is not the actual degree of the polynomial . It is just the highest exponent a term can possibly have in the polynomial . It helps us to figure out the size of array to be created for storing our polynomial .

Polynomial (const Polynomial& obj) ;

int get _degree() const ;

This method will not return the max_degree argument that we received in the constructor, but the actual degree of the polynomial, which is the value of the highest exponent among all the terms.

Polynomial& operator=(const Polynomial& other);

void add_term(int coefficient , int degree) ;

Throw an exception if the degree of the term to be added to the polynomial is greater than size of the array we created in constructor . If the term already exists in polynomial , simply overwrite the coefficient .

void remove_term (int degree);

void set_coeffecient(int coefficient , int degree) ;

friend ostream& operator<<(ostream& os, const Polynomial& poly);

Should print the polynomial on console as we usually write polynomials in maths such as **5x^4 + 3x^2 + 5**

double operator () (double x) const ;

Evaluates the value of polynomial for given value of x and returns the answer .

Polynomial operator + (const Polynomial& other) ;

Performs addition operation on polynomials and returns the new resultant polynomial . Write this function smartly to get the full credit :)

Polynomial operator * (const Polynomial& other) ;

~Polynomial () ;