Name	ADITI RAO
UID no.	202220003
Experiment No.	10

AIM:	**Programs on Abstraction: Implement Programs to demonstrate STL**

Program 1

PROBLEM STATEMENT:	Create a class Book with private data members title, author and year of publication. Write appropriate constructors for the same. Write a display() method to display a book's details and a getter method for getting a book's title. In main, create a vector to store book objects. Create a menu with the options: Add a book, Display all books, Search a book by title, Exit. Add a book should be able to add a given book to the vector you created. Display all books should be able to traverse through the vector of books created(Use an iterator for vector for the same). Search a book should be able to search for a given book through the iterator for the vector of books.

ALGORITHM:

```
class Book:
private string title
private string author
private int yearofPublication

constructor Book():
    title = ""
    author = ""
    yearofPublication = 0

constructor Book(string title, string author, int yearofPublication):
    this.title = title
    this.author = author
    this.yearofPublication = yearofPublication

method getTitle():
    return title

method displayBook():
    output "Title: " + title
    output "Author: " + author
```

```
        output "Year of Publication: " + yearofPublication

    main():
        create empty vector books
        choice = 0

        do:
            output "1. Add a book"
            output "2. Display books"
            output "3. Search a book"
            output "4. Exit"
            output "Enter your choice: "
            input choice

            switch choice:
                case 1:
                    create empty strings title, author
                    create integer yearofPublication

                    output "Enter title: "
                    ignore newline
                    input title

                    output "Enter author: "
                    input author

                    output "Enter year of publication: "
                    input yearofPublication

                    create Book object book with title, author, and yearofPublication
                    add book to books vector
                    break

                case 2:
                    if books is empty:
                        output "No books added yet."
                    else:
                        for each book in books:
                            call displayBook() method of book
                            output newline
                    break

                case 3:
```

```
                    create empty string searchTitle
                    found = false

                    output "Enter the book title: "
                    ignore newline
                    input searchTitle

                    if books is not empty:
                        for each book in books:
                            if book.getTitle() is equal to searchTitle:
                                call displayBook() method of book
                                found = true
                                break

                        if found is false:
                            output "Book not found."
                    else:
                        output "No books added yet."
                    break

                case 4:
                    output "Thank you for using the program."
                    break

                default:
                    output "Invalid choice."
                    break
            while choice is not equal to 4
```

PROGRAM:

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Book {
    private:
        string title;
        string author;
        int yearofPublication;

    public:
        Book() {
```

```
        title = "";
        author = "";
        yearofPublication = 0;
    }
    Book(string title, string author, int yearofPublication) {
        this->title = title;
        this->author = author;
        this->yearofPublication = yearofPublication;
    }

    string getTitle() {
        return title;
    }

    void displayBook() {
        cout << "Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "Year of Publication: " << yearofPublication << endl;
    }
};

int main()
{
    vector <Book> books;
    int choice;

    do {
        cout << "1. Add a book" << endl;
        cout << "2. Display books" << endl;
        cout << "3. Search a book" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {

            case 1: {
                string title, author;
                int yearofPublication;

                cout << "Enter title: ";
                cin.ignore();
                getline(cin, title);
```

```
        cout << "Enter author: ";
        getline(cin, author);
        cout << "Enter year of publication: ";
        cin >> yearofPublication;

        Book book(title, author, yearofPublication);
        books.push_back(book);
        break;
    }

    case 2: {
        for (int i = 0; i < books.size(); i++) {
            books[i].displayBook();
            cout << endl;
        }
        break;
    }

    case 3: {
        string searchTitle;
        bool found = false;

        cout << "Enter the book title: ";
        cin.ignore();
        getline(cin, searchTitle);
        if (books.size() > 0) {
            for (int i = 0; i < books.size(); i++) {
                if (books[i].getTitle() == searchTitle) {
                    books[i].displayBook();
                    found = true;
                    break;
                }
            }
            if (!found) {
                cout << "Book not found." << endl;
            }
        }
        else {
            cout << "No books added yet." << endl;
        }
        break;
    }
```

```
        case 4: {
            cout << "Thank you for using the program." << endl;
            break;
        }

        default: {
            cout << "Invalid choice." << endl;
            break;
        }
        cout << endl;
    }
  }
  while (choice != 4);

  return 0;
}
```

RESULT:

```
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 1
Enter title: More Tha Human
Enter author: Sturgeon, Theodore
Enter year of publication: 1953
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 1
Enter title: At Home in the Universe
Enter author: Duke Kauffman
Enter year of publication: 1996
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 1
Enter title: Object Oriented Programming in CPP
Enter author: Balagurusamy
Enter year of publication: 2008
```

```
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 2
Title: More Tha Human
Author: Sturgeon, Theodore
Year of Publication: 1953

Title: At Home in the Universe
Author: Duke Kauffman
Year of Publication: 1996

Title: Object Oriented Programming in CPP
Author: Balagurusamy
Year of Publication: 2008

1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 3
Enter the book title: At Home in the Universe
Title: At Home in the Universe
Author: Duke Kauffman
Year of Publication: 1996
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 3
Enter the book title: Problem Solving in CPP
Book not found.
```

```
1. Add a book
2. Display books
3. Search a book
4. Exit
Enter your choice: 4
Thank you for using the program.
```

Program 2

PROBLEM STATEMENT:	Write a program which mimics a voting system for awarding the best movie. Your program must first read the total number of votes received. Next it must read the movie names one by one. A movie name entered means a vote received for the movie. Calculate the total votes received for each movie, and output it along with the movie name. Find and print the best movie. Use a map to calculate the output. Your map should index from a string representing each movie's name to integers that store the sum of the votes for the movie.
ALGORITHM:	main(): NumberofVotes = 0

	output "Kindly Enter the number of votes:"
	input NumberofVotes
	
	create an empty map called votes
	
	output "Kindly enter movie name to resemble each vote one to a line."
	
	for i = 1 to NumberofVotes:
	  string movieName
	  input movieName
	  increment the vote count for movieName in the votes map
	
	maxVotes = 0
	bestMovie = ""
	
	for each vote in votes:
	  if vote.second > maxVotes:
	  maxVotes = vote.second
	  bestMovie = vote.first
	
	output "The best movie is " + bestMovie + " with " + maxVotes + " votes."

PROGRAM:	#include <iostream>
	#include <string>
	#include <map>
	
	using namespace std;
	
	int main()
	{
	  int NumberofVotes;
	  cout << "Kindly Enter the number of votes: " << endl;
	  cin >> NumberofVotes;
	  cin.ignore();
	
	  map<string, int> votes;
	
	  cout << "Kindly enter movie name to resemble each vote one to a line. " << endl;
	  for (int i = 0; i < NumberofVotes; i++) {
	  string movieName;
	  getline(cin, movieName);
	  votes[movieName]++;
	  }

```
    int maxVotes = 0;
    string bestMovie;

    for (auto vote : votes) {
        if (vote.second > maxVotes) {
            maxVotes = vote.second;
            bestMovie = vote.first;
        }
    }

    cout << "The best movie is " << bestMovie << " with " << maxVotes << " votes."
<< endl;
    return 0;
}
```

RESULT:

```
Kindly Enter the number of votes:
7
Kindly enter movie name to resemble each vote one to a line.
Harry Potter and the Order of the Phoenix
Harry Potter and the Order of the Phoenix
The Bourne Ultimatum
Harry Potter and the Order of the Phoenix
The Bourne Ultimatum
Wall-E
Glitter
The best movie is Harry Potter and the Order of the Phoenix with 3 votes.
```

CONCLUSION

STL (Standard Template Library) is a powerful feature in C++ that provides a collection of reusable algorithms and data structures. It includes containers (like vectors and maps), algorithms (sorting, searching), and function objects. STL enhances code efficiency, readability, and maintainability by promoting code reuse and abstracting complex operations into simple, generic functions.