**Modern Education Society’s**

**College of Engineering, Pune**

|  |  |
| --- | --- |
| **NAME OF STUDENT:** Prathamesh Kalyan Sable | **CLASS:** SE Comp 1 |
| **SEMESTER/YEAR:** Sem-3 / 2022-23 | **ROLL NO:** 015 |
| **DATE OF PERFORMANCE:**  / /2022 | **DATE OF SUBMISSION:** / /2022 |
| **EXAMINED BY:** Prof. N.R. Mhaske | **EXPERIMENT NO: C-22** |

**TITLE : TO PERFORM VARIOUS OPERATIONS ON LINK LIST .**

**PROBLEM STATEMENT :** Second year Computer Engineering class, set A of students like Vanilla ice-cream and set B of students like Butterscotch ice-cream Write C++ program to store two sets using Linked List. Compute and display-

i. Set of students who like both vanilla and butterscotch

ii. Set of students who like either vanilla or butterscotch or not both

iii. Number of students who like neither vanilla nor butterscotch

**OBJECTIVES:**

1. To understand structure of singly linked list .

2. To understand how to Create, Display and perform various operations on singly linked list.

**OUTCOMES:**

1.To analyze the problems to apply suitable algorithm and data structure .

2.To discriminate the usage of various data structures in approaching the problem solution .

3.To understand concepts of Linear Data Structure (singly linked lists ) .

**PRE-REQUISITES:**

1.Knowledge of C++ programming .

2.Knowledge of singly linked lists .

**APPARATUS:**

Working computer with g++ compiler installed.

**QUESTIONS:**

1. What is Generalized Lined List? Explain with example?

2. What are the advantages of Dynamic memory allocation?

**SOURCE CODE:**

#include <iostream>

using namespace std;

// creating node

class node

{

public:

    int data;

    node \*next;

    node()

    {

        next = NULL;

        // data = NULL;

    }

    node(int data)

    {

        this->data = data;

        this->next = NULL;

    }

};

int find\_set(node \*head, int elmt)

{

    node \*temp = new node;

    temp = head;

    int index = 0;

    while (temp != NULL)

    {

        if (temp->data == elmt)

        {

            return index;

        }

        index++;

        temp = temp->next;

    }

    return -1;

}

void delete\_set(node \*&head, int elmt)

{

    node \*temp = new node();

    temp = head;

    if (temp->data == elmt)

    {

        if (temp->next == NULL)

        {

            delete head;

            return;

        }

        else

        {

            node \*tofree = temp;

            head = temp->next;

            delete tofree;

        }

    }

    else

    {

        while (temp->next != NULL)

        {

            if (temp->next->next == NULL)

            {

                node \*current = temp->next;

                if (current->data == elmt)

                {

                    temp->next = current->next;

                    delete current;

                }

                return;

            }

            else

            {

                node \*current = temp->next;

                if (current->data == elmt)

                {

                    temp->next = current->next;

                    delete current;

                }

                temp = temp->next;

            }

        }

    }

}

void add\_set(node \*&head, int elmt)

{

    if (find\_set(head, elmt) == -1)

    {

        node \*temp = new node();

        temp = head;

        node \*newnode = new node(elmt);

        while (temp->next != NULL)

        {

            temp = temp->next;

        }

        temp->next = newnode;

    }

}

void print\_set(node \*head)

{

    node \*temp = new node;

    temp = head;

    while (temp != NULL)

    {

        cout << temp->data << " ";

        temp = temp->next;

    }

    cout << endl;

}

void delete\_index(node \*&head, int index)

{

    if (index <= -1)

    {

        return;

    }

    else if (index == 0 and head->next != NULL)

    {

        node \*tofree = head;

        head = head->next;

        delete tofree;

    }

    else

    {

        node \*temp = new node;

        temp = head;

        for (int i = 0; i < index - 1; i++)

        {

            temp = temp->next;

        }

        node \*tofree = temp->next;

        temp->next = temp->next->next;

        delete tofree;

    }

}

void get\_set(node \*&head, string flv)

{

    int n, roll;

    cout << "Enter Number of students who like " << flv << ":";

    cin >> n;

    cout << "Enter roll numbers:" << endl;

    cin >> roll;

    head = new node(roll);

    for (int i = 1; i < n; i++)

    {

        cin >> roll;

        add\_set(head, roll);

    }

}

void set\_only\_one(node \*set1, node \*set2)

{

    node \*temp = new node;

    temp = set1;

    while (temp != NULL)

    {

        if (find\_set(set2, temp->data) == -1)

        {

            cout << temp->data << " ";

        }

        temp = temp->next;

    }

    node \*temp2 = new node;

    temp2 = set2;

    while (temp2 != NULL)

    {

        if (find\_set(set1, temp2->data) == -1)

        {

            cout << temp2->data << " ";

        }

        temp2 = temp2->next;

    }

    cout << endl;

}

void set\_intersection(node \*set1, node \*set2)

{

    node \*temp = new node;

    temp = set1;

    while (temp != NULL)

    {

        if (find\_set(set2, temp->data) != -1)

        {

            cout << temp->data << " ";

        }

        temp = temp->next;

    }

    cout << endl;

}

int nothing(node \*set1, node \*set2, int total)

{

    int count = 0;

    for (int i = 1; i <= total; i++)

    {

        if ((find\_set(set1, i) == -1) and (find\_set(set2, i) == -1))

        {

            count++;

        }

    }

    return count;

}

int main()

{

    node \*seta;

    node \*setb;

    int choice, total;

    bool while\_var = true;

    cout << "Enter Total number of students :";

    cin >> total;

    get\_set(seta, "Vanilla");

    get\_set(setb, "Butterscotch");

    while (while\_var)

    {

        cout << "-- MENU --" << endl;

        cout << "1. Display both Lists" << endl;

        cout << "2. Reset Student who like vanilla" << endl;

        cout << "3. Reset Student who like Butterscotch" << endl;

        cout << "4. Set of Student who like both vanilla and butterscotch" << endl;

        cout << "5. Set of Student who like either vanilla and butterscotch, not both" << endl;

        cout << "6. Number of Student who like neither vanilla nor butterscotch" << endl;

        cout << "7. Exit" << endl;

        cout << "Enter your Choice:";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Students who like vanilla are ";

            print\_set(seta);

            cout << "Students who like Butterscotch are ";

            print\_set(setb);

            break;

        case 2:

            get\_set(seta, "Vanilla");

            break;

        case 3:

            get\_set(setb, "Butterscotch");

            break;

        case 4:

            cout << "Set of Student who like both vanilla and butterscotch are " << endl;

            set\_intersection(seta, setb);

            break;

        case 5:

            cout << "Set of Student who like either vanilla and butterscotch, not both" << endl;

            set\_only\_one(seta, setb);

            break;

        case 6:

            cout << "Number of Student who like neither are " << nothing(seta, setb, total) << endl;

            break;

        case 7:

            while\_var = false;

            break;

        default:

            cout << "Enter a valid choice." << endl;

            break;

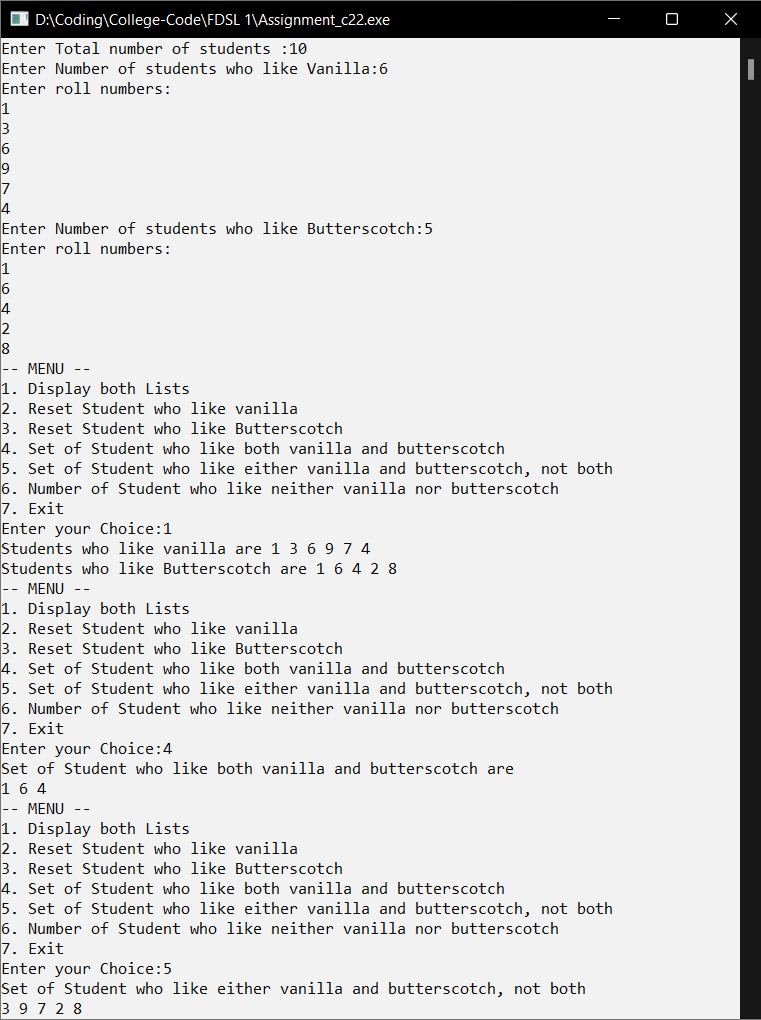
        }

    }

    return 0;

}

**OUTPUT:**

****

****