```
#pragma once
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

// node for our linked list
struct node
{
    int data; // data value (could be a lot more values)

    node *next; // we always need a "link" in a linked list
    node *previous;

    node(int x)
    { // cunstructor to make adding values easy
        data = x;
        next = NULL;
        previous = NULL;
    }
};

class MyVector
{
private:
    node *head;
    node *tail;
    int size = 0;

public:
    MyVector()
    {
        head = tail = NULL;
    }

    MyVector(string FileName)
    {
        ifstream fin;       // stream reference
                            //
        fin.open(FileName); // open the file
                            //
        fin >> size;        // get first value which contains
                            // file size
        int dat;
        // Can also use for loop since we know the exact count in file.
        // eof = end of file flag
        // `!fin.eof()` evaulates to true when we hit end of file.
        while (!fin.eof())
        {
            fin >> dat;
            pushFront(dat); // push dat onto list
```

```
            size++; // size index ++
        }
    }

    MyVector(int A[], int size)
    {
        head = NULL; // NULL = zeros
                     // and zeros imply empty

        for (int i = 0; i < size; i++)
        {
            pushFront(A[i]);
        }
    }

    MyVector(const MyVector &V1)
    {
        head = tail = NULL;
        node *temp = V1.head;
        while (temp)
        {
            pushRear(temp->data);
            temp = temp->next;
        }
    }

    void pushFront(int x)
    {
        node *temp = new node(x); // create a new node and
                                  // add data to it

        if (!head)
        { // `!head` implies empty list
            // So does `head == NULL`

            head = tail = temp; // `head = tempPtr` places addrress of
                                // tempPtr in head (points head to tempPtr)
        }
        else
        {                       // list not empty
            temp->next = head; // point tempPtr's next to what head points to
            head = temp;       // now point head to tempPtr
        }
    }

    void pushFront(MyVector V2)
    {
        node *temp = V2.tail;

        while (temp)
        {
            pushRear(temp->data);
            temp = temp->previous;
```

```
        }
    }

    void pushRear(int x)
    {
        node *temp = new node(x); // create a new node and
                                  // add data to it
        if (!head)
        { // `!head` implies empty list
            // So does `head == NULL`

            head = tail = temp; // `head = tempPtr` places addrress of
                                // tempPtr in head (points head to tempPtr)
        }
        else
        {
            tail->next = temp; // point tempPtr's next to what head points to
            tail = temp;       // now point head to tempPtr
        }
    }

    void pushRear(MyVector V2)
    {
        node *temp = V2.head;

        while (temp)
        {
            pushRear(temp->data); // print data from node
            temp = temp->next;    // move to next node
        }
    }

    int popFront()
    {
        int bruh = head->data;
        node *temp = head;
        head = head->next;
        delete temp;
        return bruh;
    }

    int popRear()
    {
        int bruh = tail->data;
        node *temp = tail;
        tail = tail->previous;
        delete temp;
        return bruh;
    }

    void PushAt(int loc, int val)
    {
        node *temp = head;
        node *newNode;
```

```
        newNode->data = val;

        int tempPos = 0; // Traverses through the list

        temp = head; // Initialize tempent to head;
        if (head != NULL)
        {
            while (temp->next != NULL && tempPos != loc)
            {
                temp->previous = temp;
                temp = temp->next;
                tempPos++;
            }
            if (loc == 0)
            {
                pushFront(val);
            }
            else if (temp->next == NULL && loc == tempPos + 1)
            {
                pushRear(val);
            }
            else if (loc > tempPos + 1)
                cout << "location does not exist" << endl;
            else
            {
                temp->next = newNode;
                newNode->next = temp;
            }
        }
    }

    void inOrderPush(int val)
    {
        node *temp = node(val);
        if (!head)
        {
            head = tail = temp;
        }
        else
        {
            if (temp->value < head->value)
            {
                //frontsert
                temp->next = head;
                head = temp;
            }
            else if (temp->value > tail->value)
            {
                //endsert
                tail->next = temp;
                tail = temp;
            }
            else
            {
```

```
                //find the location
                node *Prev = head;
                node *Curr = head;
                while (Curr->value < temp->value)
                {
                    Prev = Curr;
                    Curr = Curr->next;
                }
                Prev->next = temp;
                temp->next = Curr;
            }
        }
    }

    int popAt(int loc)
    {
        node *temp = head int pop;

        int tempPos = 0; // Traverses through the list

        // Initialize tempent to head;
        if (head != NULL)
        {
            while (temp->next != NULL && tempPos != loc)
            {
                temp->previous = temp;
                temp = temp->next;
                tempPos++;
            }
            // if(loc==0)
            // {
            //    pushFront(val);
            // }
            // else if(temp->next == NULL && loc == tempPos+1)
            // {
            //  pushRear(val);
            // }
            if (loc > tempPos + 1)
                cout << "location does not exist" << endl;
            else
            {
                pop = temp->data;
                node *temp2 = temp;
                temp = temp->previous;
                delete temp;
                return pop;
            }
        }
    }

    int find(int val)
    {
        node *temp = head;
        int i = 0;
```

```
        while (temp)
        {
            if (val == temp->data)
            {
                return i;
            }

            temp = temp->next;
            i++
        }
        return -1
    }

    void print()
    {
        node *temp = head; // temp pointer copies head

        while (temp)
        { // this loops until temp is NULL
            // same as `while(temp != NULL)`
            cout << "[";
            cout << temp->data; // print data from node
            if (temp->next)
            {
                cout << ", ";
            }
            temp = temp->next; // move to next node
        }
        cout << "]";
        cout << endl;
    }
}
}
```