/** *

- Author: Samuel Campbell
- Email: Sccampbell1019@my.msutexas.edu
- Label: P01
- Title: MyVector Class
- Course: CMPS 2143
- Semester: Fall 2021
-
- Description:

-
  ```
      Makes a My vector class that has a double linked class,
  ```

-
  ```
      and has several functions to transverse and modify the list.
  ```
-
- Usage:

-
  ```
       MyVector test;
  ```

-
  ```
       test.function
  ```
-
- Files: in1.dat ***/

#include #include #include

using namespace std;

struct Node { // struct containing the basic variables needed for int data; // list traversal and modification Node *next; Node *prev; Node(int x) { data = x; next = NULL; } };

/**

- Class MyVector
-
- Description:

-
  ```
    contains the constructor and functions to modify the linked list
  ```

-
- Public Methods:

-
  ```
  -             MyVector()
  ```

-
  ```
  -             MyVector(int* Arr, int size)
  ```

-
  ```
   -             MyVector(string filename)
  ```

-
  ```
   - void        PushFront(int val)
  ```

-
  ```
   - void        PushFront(MyVector V2)
  ```

-
  ```
   - void        PushFront(int val)
  ```

-
  ```
   - void        PushRear(int val)
  ```

-
  ```
   - void        PushRear(MyVector V2)
  ```

-
  ```
   - int         popAt(int x)
  ```

-
  ```
   - bool        PushAt(int index, int val)
  ```

-
  ```
   - int         PopFront()
  ```

-
  ```
   - int         PopRear()
  ```

-
  ```
   - int         FindAt(int val)
  ```

-
  ```
   - void        Print()
  ```

-
  ```
   -             ~MyVector()
  ```
-
- Usage:
-

-
  ```
  MyVector test("in1.dat");
  ```

-
  ```
  MyVector test;
  ```

-
  ```
  MyVector test2;
  ```

-
  ```
  test.PushRear(19);
  ```

-
  ```
  test2.PushRear(10)
  ```

-
  ```
  test.Print();
  ```
-
-

*/ class MyVector { private: Node *head; Node *tail; int size;

public: /** * Public : MyVector * * Description: * Default Constructor * * Params: * * * Returns: * none */ MyVector() { head = NULL; tail = NULL; size = 0; }

/** * Public : MyVector * * Description: * constructor reads array and pushes into back of list

```
*
* Params:
*     int*  Arr
*     int   size
*
* Returns:
*     none
*/
```

MyVector(int *Arr, int size) { head = NULL; tail = NULL; size = 0;

```
for (int i = 0; i < size; i++) {
  PushRear(Arr[i]);
}
```

}

/** * Public : MyVector(string filename) * * Description: * constructor using data from file

```
  *
  * Params:
  *     string filename

  *
  * Returns:
  *     none
  */
```

MyVector(string filename) { head = NULL; tail = NULL; size = 0;

```
ifstream fin;
int x;
fin.open(filename);
while (!fin.eof()) {
  fin >> x;
  PushRear(x);
}
```

}

/**

- Public : PushFront

-
- Description:
- Takes in int val and pushes to the front of the list
-
- Params:

-
  ```
  int val
  ```
-
- Returns:

-
  ```
  Void
  ```

*/ void PushFront(int val) { Node *Temp = new Node(val); if (head == NULL) { head = Temp; tail = head; size++; } else { Temp->next = head; head = Temp; size++; } // cout<<"tail: "<data<<endl; //cout<<"tail&:" <<tail<<endl; }

/** * Public : PushFront * * Description: * Takes in int val and pushes to the front of the list

```
*
* Params:
*     MyVector V2

*
* Returns:
*     Void
*/
void PushFront(MyVector V2){
  int v;
  while((!V2.Empty())){
    v = V2.PopRear();
    cout<<v<<endl;
    PushFront(v);
}
cout<<"test"<<endl;
}
```

/** * Public : Empty * * Description: *

```
*checks if the list is empty
* Params:
*     none

*
* Returns:
*     bool
*/
bool Empty(){
  return size == 0;
}
```

/** * Public : PushRear * * Description: * Takes in int val and pushes to the back of the list

```
*
* Params:
*     int val

*
* Returns:
*     Void
*/
```

void PushRear(int val) { Node *Temp = new Node(val);

```
// empty list set everything = to the new node.
if (head == NULL) {
  head = Temp;
  tail = head;
  size++;
} else {
  tail->next = Temp;
  tail = Temp;
  size++;
}
```

}

/**

- Public : PushRear
-
- Description:

- Takes in MyVector and pushes to the back of the list
-
- Params:

-
  ```
  MyVector V2
  ```
-
- Returns:

-
  ```
  Void
  ```

*/ void PushRear(MyVector V2) { int x = V2.PopFront(); while (x != -1) { PushRear(x); x = V2.PopFront(); size++;

```
}
```

}

/** * Public : PopAt * * Description: * takes in a index and pops at that location in list(use size for this then loop index amount of times then place node there)

```
 *
 * Params:
 *     int size

 *
 * Returns:
 *      value if the node at the said index
 */
 // DONT FORGET ABOUT TAIL
```

int popAt(int x) { if (x >= size) { return -1; } else { Node *prev = NULL; Node *temp = head; int loc = 0; while (loc != x) { prev = temp; temp = temp->next; loc++; } prev->next = temp->next; int cont = temp->data; delete temp; size--; return cont; } }

/** * Public : PushAt * * Description: * Pushes a node carying a value at a certain index takes in a index and puts at that location in list(use size for this then loop index amount of times then place node there) * * Params: * int index int val * * Returns: * bool : whether it can successfuly enter in a value */

bool PushAt(int index, int val) { Node *prev = head; // get previous and next pointers Node *current = head; Node *nNode = new Node(val); // needed ne memory for new value

```
  while (index > 0) {
    prev = current;
    current = current->next;
    index--;
  }
  cout << prev->data << "," << current->data << endl;
  prev->next = nNode;         // Need to point prev (next) to the new memory.
  nNode->next = current;      // Need to point nNode's next to current.

  size++;
  return true;
```

}

/** * Public : PopFront * * Description: * Pops front value from list * * Params: * None * * Returns: * int : value at front */ int PopFront() { if (head == NULL) { return -1; } else { int value = head->data; Node *Temp = head; head = head->next; delete Temp; size--; return value; } }

/** * Public : PopRear * * Description: * Pops rear value from list * * Params: * None * * Returns: * int : value at rear */ int PopRear() { int data; cout<<"size:"<<size<<endl; if (head == NULL) { // empty list return sentinel value

```
    return -1;
  } else {

   if (tail==head){       // dealing with a list with one node
     data=tail->data;
     cout<<"ddata:"<<data<<endl;
     cout<<"deleting head "<<head<<endl;
     //delete head;
     head=tail=NULL;
     size = 0;

   }else{                  // dealing with a list with multiple nodes


      Node *prev = head;

      while (prev->next != tail) {
        prev = prev->next;
      }
      cout<<"tail: "<<tail<<endl;
      cout<<"head: "<<head<<endl;
      cout<<"prev: "<<prev<<endl;

      Node *temp = tail;
      data = tail->data;
      tail = prev;
      tail->next = nullptr;

      delete temp;
      size--;
    }
    return data;
  }
```

}

/** * Public : Find * * Description: * trys to see if the value is in list if not return -1 * * Params: * int val * * Returns: * int : index */ int Find(int val) { Node *current = head; int size = 0; while (current != NULL) { current = current->next; size++; } current = head; for (int i = 0; i < size; i++) { if (current->data == val) { cout << val << " found at index: " << i << endl; return -1; } else current = current->next; } cout << val << " not found" << endl; return -1; }

/** * Public : Print * * Description: * prints the list * * Params: * none * * Returns: * void */ void Print(ofstream &fout) { // ofstream fout; // fout.open("outfile.txt");

```
  Node *Temp = head;
  while (Temp != NULL) {
    // cout << Temp->data << "--Curr  value--"<<"\n";
    cout << Temp->data << "->";
    fout << Temp->data << "->";
    // fout << "hi";

    Temp = Temp->next;
  }
  cout << endl;
  fout << endl;
```

}

~MyVector() { Node *curr = head; Node *prev = head; while (curr) { prev = curr; curr = curr->next; delete prev; } delete curr; } };

int main() { MyVector test("input.dat"); ofstream fout; fout.open("outfile.txt"); fout<< "V1: "; test.Print(fout); MyVector test2; test2.PushFront(12); test2.PushFront(13); test2.PushFront(14); test2.PushRear(99); fout << "V2: "; test2.Print(fout); test.PushRear(10); test.PushRear(20); test.PushRear(30); test.PushRear(40); test.PushAt(2, 2); fout<< "V1: "; test.Print(fout); test.popAt(2); fout<<"combined: "; test.PushFront(test2);

test.Print(fout); fout.close(); return 0; }