CSCI 140 PA 3 Submission

Due Date: 09/09/2021 Late (date and time):________________

Name(s): Nero Li

Exercise 1 -- need to submit source code and I/O

-- check if completely done ✓; otherwise, discuss issues below

Source code below:

```
/*  Program: PA_3_exercise_1
    Author: Nero Li
    Class: CSCI 220
    Date: 09/09/2021
    Description:
        Provide an array implementation to maintain a
        list of names. The following operations are
        available: insert rear, insert front, remove
        a particular element, and print the whole list.

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>

using namespace std;

string arr[10];
size_t count{0};

void insertRear(string name)
```

```
{
    arr[count++] = name;
}

void insertFront(string name)
{
    ++count;
    for (size_t i = 1; i < count; ++i)
    {
        arr[i] = arr[i - 1];
    }
    arr[0] = name;
}

void removeItem(string name)
{
    for (size_t i = 0; i < count; ++i)
    {
        if (arr[i] == name)
        {
            for (size_t j = i; j < count; ++j)
            {
                arr[j] = arr[j + 1];
            }
            --count;
            return;
        }

    }
}

void print()
{
    for (size_t i = 0; i < count; ++i)
    {
        cout << arr[i] << ' ';
    }
    cout << endl;
}

int main()
{
    insertFront("Jo");
    insertFront("Jane");
    insertRear("John");
    insertRear("Kim");
    removeItem("Jo");
    print();
    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
Jane John Kim
Author: Nero Li
```

Exercise 2 -- need to submit source code and I/O

-- check if completely done ✓; otherwise, discuss issues below

Source code below:

```
/*  Program: PA_3_exercise_2
    Author: Nero Li
    Class: CSCI 220
    Date: 09/09/2021
    Description:
        Provide a linked list implementation to maintain a
        list of names. The following operations are
        available: insert rear, insert front, remove
        a particular element, and print the whole list.

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>

using namespace std;

struct Node
{
    string str;
    Node *next = NULL;
};

Node *head{NULL};

void insertRear(string name)
{
    Node *cur{head};
    while (cur->next != NULL)
    {
        cur = cur->next;
    }
    cur->next = new Node;
    cur->next->str = name;
    cur->next->next = NULL;
}

void insertFront(string name)
{
    Node *cur = new Node;
    cur->str = name;
    cur->next = head;
    head = cur;
}

void removeItem(string name)
{
    Node *cur{head};
    if (cur->str == name)
    {
        head = cur->next;
        delete cur;
        return;
    }

    while (cur->next != NULL)
    {
        if (cur->next->str == name)
        {
            Node *old = cur->next;
            cur->next = old->next;
            delete old;
            return;
        }
        cur = cur->next;
    }
}

void print()
{
    Node *cur{head};
    while (cur != NULL)
    {
        cout << cur->str << ' ';
        cur = cur->next;
    }
    cout << endl;
}

int main()
{
    insertFront("Jo");
    insertFront("Jane");
    insertRear("John");
    insertRear("Kim");
    removeItem("Jo");
    print();
    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
Jane John Kim
Author: Nero Li
```

Answer for Question 1:

In my opinion, linked list is easier to implement than array.

For insert new string in front of the queue, you need a for-loop to move everything forward and get a space for your new string to get inside. For linked list, just create a new node, change the next address to the current head, and change the head, and everything will be set.

For insert new string at the end of the queue, you can add a count number to directly add a new string after the queue. However, it requires you to create a new variable to store your count integer. For linked list, you need to go to the end of the queue and add a new node after the end node. This time linked list will need to use a for-loop to implement the function, but linked list won't need to have a count number since we just need to check the next pointer is NULL or not.

For remove a string, array require you to go through the whole queue in order to move each element backward. For linked list when you find the node you want to remove, you can just move it out of the queue by chance the next pointer and delete the node.

For print out the queue, it has similar situation to the insert new string at the end. However, this time they all need to go through the whole queue to print all the elements, so that means array need one more thing to implement this function, which is the count integer.

As a result, linked list has three functions works better than array, so linked list is easier to implement than the array.

Answer for Question 2:

One function would be more efficient, and one function wouldn't. Insert a new node in front of the queue won't have a big change, but for inserting a new string back, you can use your head Node's previous pointer to add your new node. For remove a node, you cannot just change the next pointer as before. Instead, you also need to change your previous pointer to make sure it won't point to a NULL address. Finally, for print out the queue, it won't have big difference just as the function for insert a new node in front of the queue. As a result, it is more flexible than a simple linked list, but for some times, we will need to modify more variables in order to make it work properly.

Extra Credit 1

Source code below:

```
/*  Program: PA_3_extra_1
    Author: Nero Li
    Class: CSCI 220
    Date: 09/09/2021
    Description:
        Add recPrint() operation to exercise 1 to print
        the list using a recursive function/method.

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>

using namespace std;

string arr[10];
size_t count{0};

void insertRear(string name)
{
    arr[count++] = name;
}

void insertFront(string name)
```

```
{
    ++count;
    for (size_t i = 1; i < count; ++i)
    {
        arr[i] = arr[i - 1];
    }
    arr[0] = name;
}

void removeItem(string name)
{
    for (size_t i = 0; i < count; ++i)
    {
        if (arr[i] == name)
        {
            for (size_t j = i; j < count; ++j)
            {
                arr[j] = arr[j + 1];
            }
            --count;
            return;
        }

    }
}

void print()
{
    for (size_t i = 0; i < count; ++i)
    {
        cout << arr[i] << ' ';
    }
    cout << endl;
}
```

```
void recPrint(string arr[], size_t i)
{
    if (i != 0)
    {
        recPrint(arr, i - 1);
    }

    cout << arr[i] << ' ';

    if (i == count)
    {
        cout << endl;
    }
}

int main()
{
    insertFront("Jo");
    insertFront("Jane");
    insertRear("John");
    insertRear("Kim");
    removeItem("Jo");
    recPrint(arr, count);
    //print();
    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
Jane John Kim
Author: Nero Li
```

Extra Credit 2

Source code below:

```
/*  Program: PA_3_extra_2
    Author: Nero Li
    Class: CSCI 220
    Date: 09/09/2021
    Description:
        Add recPrint() operation to exercise 2 to print
        the list using a recursive function/method.

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>

using namespace std;

struct Node
{
    string str;
    Node *next = NULL;
};

Node *head{NULL};

void insertRear(string name)
{
    Node *cur{head};
    while (cur->next != NULL)
    {
        cur = cur->next;
    }
    cur->next = new Node;
    cur->next->str = name;
    cur->next->next = NULL;
}

void insertFront(string name)
{
    Node *cur = new Node;
    cur->str = name;
    cur->next = head;
    head = cur;
}

void removeItem(string name)
{
    Node *cur{head};
    if (cur->str == name)
    {
        head = cur->next;
        delete cur;
        return;
    }

    while (cur->next != NULL)
    {
        if (cur->next->str == name)
        {
            Node *old = cur->next;
            cur->next = old->next;
            delete old;
            return;
        }
        cur = cur->next;
    }
}
```

```
void print()
{
    Node *cur{head};
    while (cur != NULL)
    {
        cout << cur->str << ' ';
        cur = cur->next;
    }
    cout << endl;
}

void recPrint(Node *cur)
{
    cout << cur->str << ' ';
    if (cur->next != NULL)
    {
        recPrint(cur->next);
    }
    else
    {
        cout << endl;
    }
}

int main()
{
    insertFront("Jo");
    insertFront("Jane");
    insertRear("John");
    insertRear("Kim");
    removeItem("Jo");
    recPrint(head);
    //print();
    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
Jane John Kim
Author: Nero Li
```