Mapúa University

School of EECE

Department of Computer Engineering

Lab 2- Linked List and STL Vectors

Santos, Joaquin S. II

2018130832

CPE104L (Data Structures and Algorithms)

C2

II. Answers to Questions

1. **Briefly discuss the advantage of linked list over arrays.**
Unlike arrays, a linked list can be used if the size is unknown for a certain data. Arrays are only limited to what the programmer inputs. Let's say the size of the array is 10, this size cannot increase. Unlike a linked list, the size can still be adjusted. And overall, a linked list uses lesser memory than an array.

2. **Given the linked list shown in Figure 2.2 , answer the following questions:**
   a. **What is the output of each of the following C++ statements?**

Statement	Output
`cout<<list->info;`	Prints out the data of info located in list
`cout<<A->info;`	Prints out data from node A
`cout<<list->link->link->info;`	It prints data which are 2 nodes away from list

   b. **What is the output of the following C++ code?**

```
p = list;
while(p!=NULL)
  cout<< p->info <<" ";
  p = p-> link;
cout<<endl;
```

Output
Prints all the nodes which are located at p. Wherein p=list.

   c. **If the code snippet below is valid, show the output. If it is invalid, explain why.**

```
p = A;
p = p -> link;
s = p;
p-> link = NULL;
s = s-> link;
cout<< p->info <<" "
  << s->info << endl;
```

Output
This will not have an output due to an error. Since s=NULL and the variable s=s->NULL will still give out a NULL pointer error.

3. **Complete the Type and Description of STL containers shown in Table 2.2**

Table 2.2		
Container	**Type**	**Description**
Vector	Sequential	Dynamic array
Deque	Queue	Double-ended queue
List	Container	Inset and remove items
Map	Associative container	In the map, keys act as the index
Multimap	Associative container	Located in the standard template library of C++
Multiset	Associative container	Can have some values
Priority_queue	Derived container	It contains high priority elements
Set	Associative container	Has a set of unique objects
Stack	Container adaptor	It operated within the context of LIFO

4. **Write the strengths and weaknesses of vectors in the space provided**

Advantages:

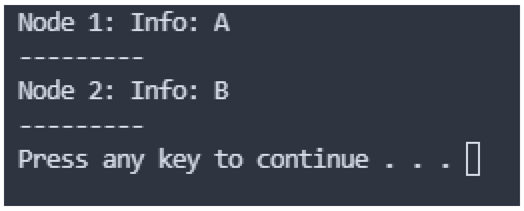
The sizes of vectors are modifiable and is also efficient when doing insertion and deletion. The elements of a Vector can also be deleted unlike an array where you can't.

Disadvantage:

A vector is an object. This means that more memory consumption is to be expected when using these.

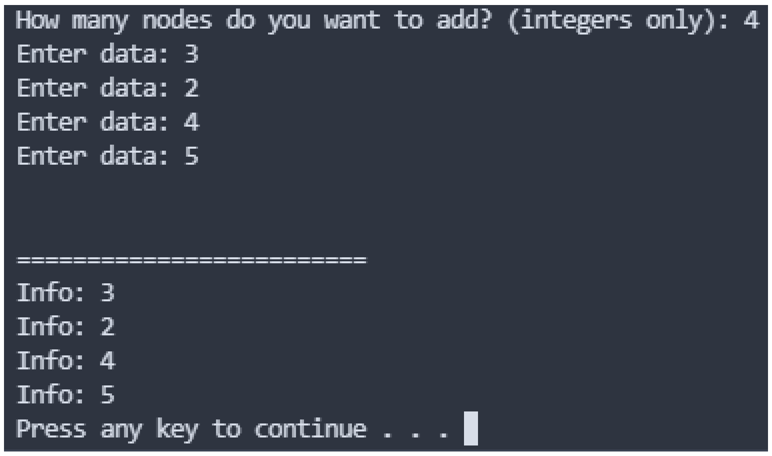
III. Screenshot of Experiment Outputs

Program Listing 2.1



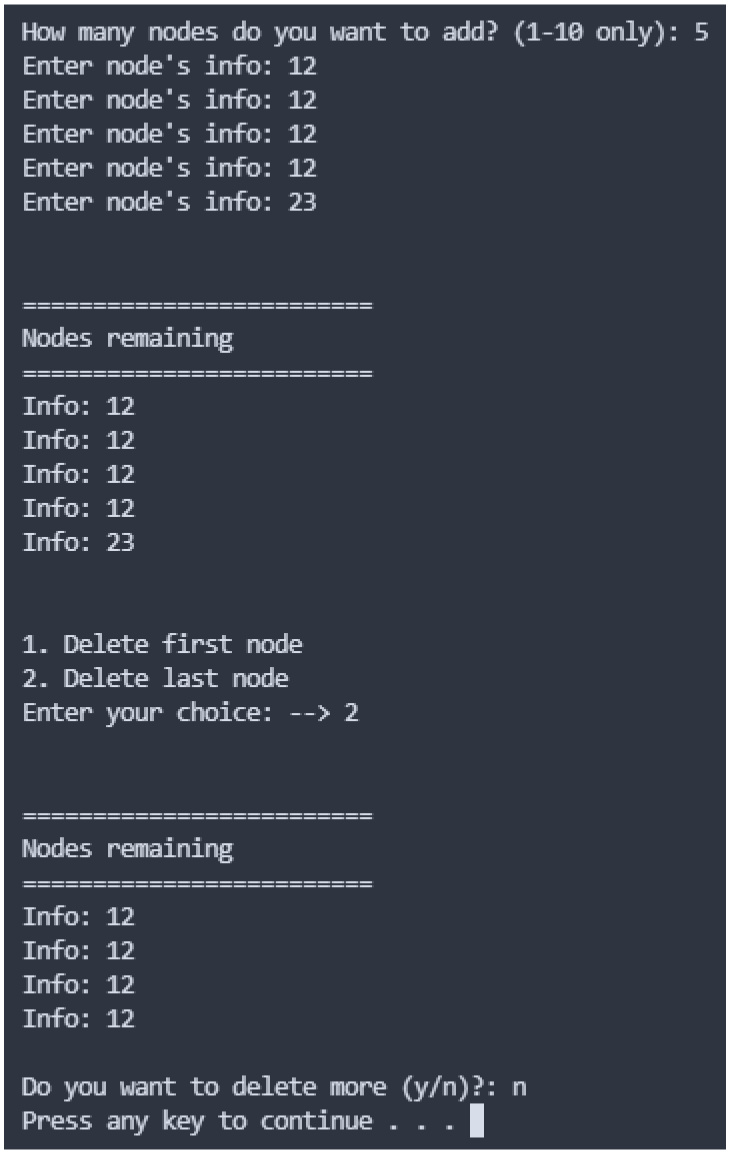
```
Node 1: Info: A
---------
Node 2: Info: B
---------
Press any key to continue . . .
```

Program Listing 2.2



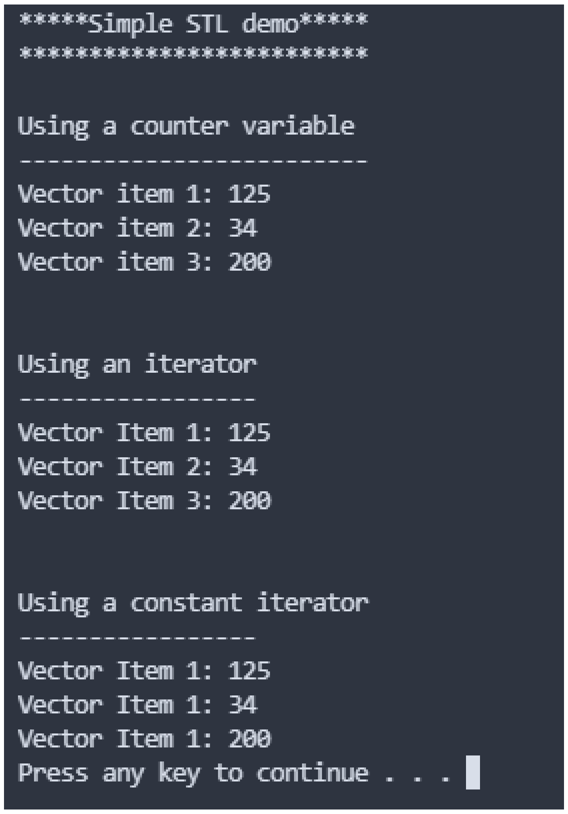
```
How many nodes do you want to add? (integers only): 4
Enter data: 3
Enter data: 2
Enter data: 4
Enter data: 5


=========================
Info: 3
Info: 2
Info: 4
Info: 5
Press any key to continue . . .
```

Program Listing 2.3



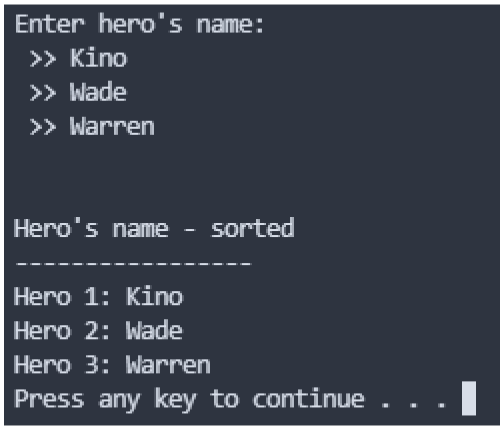
```
How many nodes do you want to add? (1-10 only): 5
Enter node's info: 12
Enter node's info: 12
Enter node's info: 12
Enter node's info: 12
Enter node's info: 23


=========================
Nodes remaining
=========================
Info: 12
Info: 12
Info: 12
Info: 12
Info: 23


1. Delete first node
2. Delete last node
Enter your choice: --> 2


=========================
Nodes remaining
=========================
Info: 12
Info: 12
Info: 12
Info: 12

Do you want to delete more (y/n)?: n
Press any key to continue . . .
```

Program Listing 2.4



```
*****Simple STL demo*****
*************************

Using a counter variable
-------------------------
Vector item 1: 125
Vector item 2: 34
Vector item 3: 200


Using an iterator
-----------------
Vector Item 1: 125
Vector Item 2: 34
Vector Item 3: 200


Using a constant iterator
-----------------
Vector Item 1: 125
Vector Item 1: 34
Vector Item 1: 200
Press any key to continue . . .
```

Program Listing 2.5



```
Enter hero's name:
 >> Kino
 >> Wade
 >> Warren


Hero's name - sorted
-----------------
Hero 1: Kino
Hero 2: Wade
Hero 3: Warren
Press any key to continue . . .
```

IV. Answer to Program Exercise

Create a header and source files that contain the class definition and implementation of the following members:

a. Define studentName (string), studentId(string), and grade(float) data members.

b. Define three constructors to initialize the data members.

c. Define member methods to implement the creation, insertion (could be anywhere), and deletion (could be anywhere) of student records implemented as linked list.

Name the source file that contains the main function as ProgExer03.cpp. You may choose an appropriate name for the class header and source files.

studentInfo.h

```
#ifndef studentInfo_H
#define studentInfo_H

#include <iostream>

class Node {
    public:
        std::string studentName;
        float grade;
        int studentId;
    Node * next;
    void print_student(Node * n);
    void setStudentInfo(std::string,int,float);
};

#endif
```

studentInfoImp.cpp

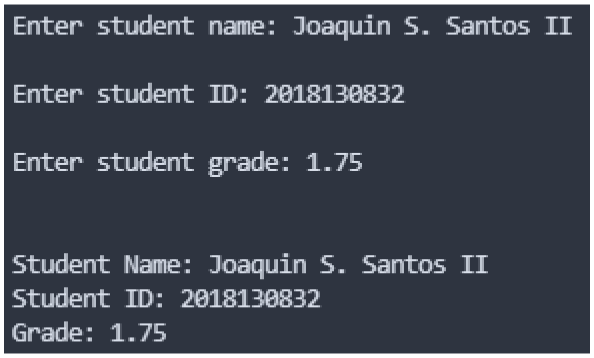
```
#include "studentInfo.h"
#include <iostream>


void Node::print_student(Node * n)
{
    std::cout <<"Student Name: "<<n->studentName << std::endl;
    n = n->next;
    std::cout <<"Student ID: "<<n->studentId << std::endl;
    n = n->next;
    std::cout <<"Grade: "<<n->grade << std::endl;
}
```

```cpp
void Node::setStudentInfo(std::string name, int ID, float grade)
{
    Node * head = NULL;
    Node * second = NULL;
    Node * third = NULL;

    head = new Node();
    second = new Node();
    third = new Node();

    head->studentName = name;
    head->next = second;

    second->studentId = ID;
    second->next = third;

    third->grade = grade;
    third->next = NULL;

    Node::print_student(head);
}
```

ProgExer03.cpp

```cpp
#include <iostream>
#include <string>
#include "studentInfo.h"



int main() {
    Node Node;
    std::string studentName;
    int studentID;
    float grade;
    std::cout<<"\nEnter student name: ";
    std::getline(std::cin,studentName);
    std::cout<<"\nEnter student ID: ";std::cin>>studentID;
    std::cout<<"\nEnter student grade: ";std::cin>>grade;
    std::cout<<"\n\n";
    Node.setStudentInfo(studentName,studentID,grade);
}
```

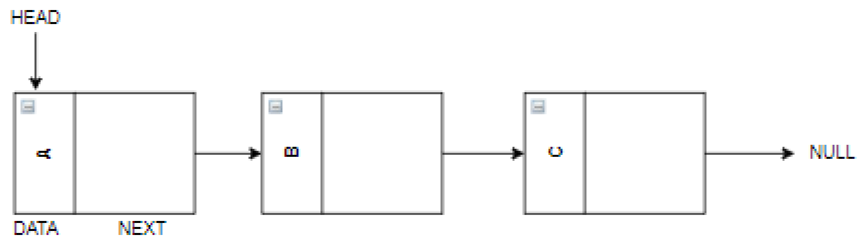
Sample Output:



```
Enter student name: Joaquin S. Santos II

Enter student ID: 2018130832

Enter student grade: 1.75


Student Name: Joaquin S. Santos II
Student ID: 2018130832
Grade: 1.75
```

V. Discussion of Report

A linear data structures wherein its elements are not located in contiguous memory locations are called a linked list. By using pointers, the elements of a linked list can be linked. Take this chart for example



As we can see, there are node A, B, and C. these nodes contain data and are linked with each other. Each node contains a data and a reference to the node next to it.

A Vector can be seen in STL or Standard Template Library on C++. These vectors are made up by sequence containers which stores elements. These are usually used with dynamic data. Unlike arrays, the sizes can be modified. They can increase or decrease in size. The elements of a vector can also be deleted. This means that vectors can be useful if the data size is unknown. Although since vectors are objects, they consume more memory.

VI. Conclusion

As we can see, linked lists can have nodes which contain different data types and vectors are also like arrays but is more flexible. Therefore, we can conclude that using linked lists and vectors make your program more flexible to the situation. Although it may use more memory compared to using arrays.