Lab 7 Pre-Lab Submission

Michael Brodskiy

Professor: S. Shazli

March 23, 2023

```
bash-4.4$ gdb
GNU gdb (GDB) Red Hat Enterprise Linux 8.2-18.el8
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file person
Reading symbols from person...done.
(gdb) start
Temporary breakpoint 1 at 0x400b2d: file person.cpp, line 26.
Starting program: /Users/Student/mbrodskiy/person

Temporary breakpoint 1, main () at person.cpp:26
26              Person person;
Missing separate debuginfos, use: yum debuginfo-install glibc-2.28-211.el8.x86_64
86_64 libstdc++-8.5.0-16.el8_7.x86_64
(gdb) next
27              person.name = "John";
(gdb) next
28              person.age = 10;
(gdb) next
29              PrintPerson(&person);
(gdb) step
PrintPerson (person=0x7fffffffe5d0) at person.cpp:18
18      cout << person->name << " is " << person->age << " years old\n";
(gdb) print person.name
$1 = "John"
(gdb) next
John is 10 years old
20      }
(gdb) print person.age
$2 = 10
(gdb) next
main () at person.cpp:26
26              Person person;
(gdb) next
31      }
(gdb) next
0x00007ffff7114d85 in __libc_start_main () from /lib64/libc.so.6
(gdb) next
Single stepping until exit from function __libc_start_main,
which has no line number information.
[Inferior 1 (process 2241569) exited normally]
(gdb)
```

Figure 1: `gdb` output

The `gdb` commands may be explained as follows:

- `file person` selects the binary called person as the file for analysis
- `start` begins analysis of "person"
- `next` moves to the next point of interest
- `step` enters the function at the current line
- `print` prints the known information for a certain, specified value

Listing 1: Menu Printing Code

```
#include <iostream>
#include <string>
using namespace std;

// Linked List Management Code
struct Person
{
        // Unique identifier for the person
        int id;
        // Information about person
        string name;
        int age;
        // Pointer to next person in list
        Person *next;
};
struct List
{
        // First person in the list. A value equal to NULL
           indicates that the
        // list is empty.
        Person *head;
        // Current person in the list. A value equal to NULL
           indicates a
        // past-the-end position.
        Person *current;
        // Pointer to the element appearing before 'current'. It
           can be NULL if
        // 'current' is NULL, or if 'current' is the first element
            in the list.
        Person *previous;
        // Number of persons in the list
        int count;
};
```

```
30
31 // Give an initial value to all the fields in the list.
32 void ListInitialize(List *list)
33 {
34         list->head = NULL;
35         list->current = NULL;
36         list->previous = NULL;
37         list->count = 0;
38 }
39 // Move the current position in the list one element forward. If
       last element
40 // is exceeded, the current position is set to a special past-the-
       end value.
41 void ListNext(List *list)
42 {
43         if (list->current)
44         {
45                 list->previous = list->current;
46                 list->current = list->current->next;
47         }
48 }
49 // Move the current position to the first element in the list.
50 void ListHead(List *list)
51 {
52         list->previous = NULL;
53         list->current = list->head;
54 }
55 // Get the element at the current position, or NULL if the current
       position is
56 // past-the-end.
57 Person *ListGet(List *list)
58 {
59         return list->current;
60 }
61 // Set the current position to the person with the given id. If no
       person
62 // exists with that id, the current position is set to past-the-
       end.
63 void ListFind(List *list, int id)
64 {
65         ListHead(list);
66         while (list->current && list->current->id != id)
67                 ListNext(list);
68 }
69 // Insert a person before the element at the current position in
```

```
   the list. If
// the current position is past-the-end, the person is inserted at the end of
// the list. The new person is made the new current element in the list.
void ListInsert(List *list, Person *person)
{
        // Set 'next' pointer of current element
        person->next = list->current;
        // Set 'next' pointer of previous element. Treat the special case where
        // the current element was the head of the list.
        if (list->current == list->head)
                list->head = person;
        else
                list->previous->next = person;
        // Set the current element to the new person
        list->current = person;
}
// Remove the current element in the list. The new current element will be the
// element that appeared right after the removed element.
void ListRemove(List *list)
{
        // Ignore if current element is past-the-end
        if (!list->current)
                return;
        // Remove element. Consider special case where the current element is
        // in the head of the list.
        if (list->current == list->head)
                list->head = list->current->next;
        else
                list->previous->next = list->current->next;
        // Free element, but save pointer to next element first.
        Person *next = list->current->next;
        delete list->current;
        // Set new current element
        list->current = next;
}
void PrintPerson(Person *person)
{
        cout << "Person with ID: " << person->id << endl;
        cout << "\tName: " << person->name << endl;
        cout << "\tAge: " << person->age << endl << endl;;
```

```
109 }
110
111 /** main function: Will create and process a linked list
112  */
113 int main() {
114         List list;                                  // Create the main
                list
115         ListInitialize(&list);                      // Initialize the
              list
116 //*************** PUT THE REST OF YOUR CODE HERE
       *****************
117
118         string options[] = {"Add a person", "Find a person", "
              Remove a person", "Print the list", "Exit"};
119
120         int choice = 0;
121
122         do {
123
124                 for (int i = 0; i < 5; i++) {
125
126                         cout << (i + 1) << ". " << options[i] <<
                              endl;
127
128                 }
129
130                 cout << "Select an option: ";
131                 cin >> choice;
132                 cout << "You selected: ";
133
134                 if (choice == 1) {
135
136                         cout << "\"" << options[choice - 1] << "\"
                              " << endl;
137
138                 }
139
140                 else if (choice == 2) {
141
142                         cout << "\"" << options[choice - 1] << "\"
                              " << endl;
143
144                 }
145
146                 else if (choice == 3) {
```

```
147
148                         cout << "\"" << options[choice - 1] << "\"
                               " << endl;
149
150                 }
151
152                 else if (choice == 4) {
153
154                         cout << "\"" << options[choice - 1] << "\"
                               " << endl;
155
156                 }
157
158                 else if (choice == 5) {
159
160                         cout << "\"" << options[choice - 1] << "\"
                               " << endl;
161
162                 }
163
164                 else {
165
166                         cout << "Error. Invalid option. Try again.
                               " << endl << endl;
167
168                 }
169
170         } while (choice < 1 || choice > 5);
171
172
173
174
175
176
177
178
179 } //end main
```

```
bash-4.4$ ./personList
1. Add a person
2. Find a person
3. Remove a person
4. Print the list
5. Exit
Select an option: 0
You selected: Error. Invalid option. Try again.

1. Add a person
2. Find a person
3. Remove a person
4. Print the list
5. Exit
Select an option: 4
You selected: "Print the list"
bash-4.4$
```

Figure 2: Sample menu output