

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=0x7fffffff5d0) 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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Linked List Management Code
6 struct Person
7 {
8     // Unique identifier for the person
9     int id;
10    // Information about person
11    string name;
12    int age;
13    // Pointer to next person in list
14    Person *next;
15 };
16 struct List
17 {
18     // First person in the list. A value equal to NULL
19     // indicates that the
20     // list is empty.
21     Person *head;
22     // Current person in the list. A value equal to NULL
23     // indicates a
24     // past-the-end position.
25     Person *current;
26     // Pointer to the element appearing before 'current'. It
27     // can be NULL if
28     // 'current' is NULL, or if 'current' is the first element
29     // in the list.
30     Person *previous;
31     // Number of persons in the list
32     int count;
33 };
```

```
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
70 // the current position is past-the-end, the person is inserted at
    the end of
71 // the list. The new person is made the new current element in the
    list.
72 void ListInsert(List *list, Person *person)
73 {
74     // Set 'next' pointer of current element
75     person->next = list->current;
76     // Set 'next' pointer of previous element. Treat the
        special case where
77     // the current element was the head of the list.
78     if (list->current == list->head)
79         list->head = person;
80     else
81         list->previous->next = person;
82     // Set the current element to the new person
83     list->current = person;
84 }
85 // Remove the current element in the list. The new current element
    will be the
86 // element that appeared right after the removed element.
87 void ListRemove(List *list)
88 {
89     // Ignore if current element is past-the-end
90     if (!list->current)
91         return;
92     // Remove element. Consider special case where the current
        element is
93     // in the head of the list.
94     if (list->current == list->head)
95         list->head = list->current->next;
96     else
97         list->previous->next = list->current->next;
98     // Free element, but save pointer to next element first.
99     Person *next = list->current->next;
100     delete list->current;
101     // Set new current element
102     list->current = next;
103 }
104 void PrintPerson(Person *person)
105 {
106     cout << "Person with ID: " << person->id << endl;
107     cout << "\tName: " << person->name << endl;
108     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
115         list
116     ListInitialize(&list);                    // Initialize the
117         list
118     /******* PUT THE REST OF YOUR CODE HERE
119         *****/
120
121     string options[] = {"Add a person", "Find a person", "
122         Remove a person", "Print the list", "Exit"};
123
124     int choice = 0;
125
126     do {
127
128         for (int i = 0; i < 5; i++) {
129
130             cout << (i + 1) << ". " << options[i] <<
131                 endl;
132
133         }
134
135         cout << "Select an option: ";
136         cin >> choice;
137         cout << "You selected: ";
138
139         if (choice == 1) {
140
141             cout << "\"" << options[choice - 1] << "\"
142                 << endl;
143
144         }
145
146         else if (choice == 2) {
147
148             cout << "\"" << options[choice - 1] << "\"
149                 << endl;
150
151         }
152
153         else if (choice == 3) {
```

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