Worksheet 17 - Group 1

Worksheet Group 1 Members

Marc Clinedinst: clinedim@onid.oregonstate.edu
Kelby Faessler: faesslek@onid.oregonstate.edu
James Fitzwater: fitzwatj@onid.oregonstate.edu
Tom Gariepy: gariepyt@onid.oregonstate.edu
Sean Reilly: reillys@onid.oregonstate.edu
Joseph Struth: struthj@onid.oregonstate.edu

Worksheet 17: Linked List Introduction, List Stack

In this worksheet, we implement four functions that define the behavior of the stack data structure when it is implemented on top of a linked list. More specifically, we implement the `linkedListStackPush`, `linkedListStackTop`, `linkedListStackPop`, and `linkedListStackIsEmpty` functions. The code for these functions is provided below, with accompanying comments where extra explanation is necessary.

```
/*
     This function pushes (adds) a value to the top of the stack.  The function first allocates
     memory for a new link and checks that the memory was successfully allocated.  The function
     then configures the value and next fields for the link.  Specifically, it sets value to d,
     and makes next point to newLink.  firstLink is then set to point to newLink.  This in
     effect makes the new element the topmost element.
*/
void linkedListStackPush(struct linkedListStack *s, TYPE d) {
     struct link *newLink = (struct link *) malloc(sizeof(struct link));
     assert(newLink != 0);
     newLink->value = d;
     newLink->next = s->firstLink;
     s->firstLink = newLink;
```

```
}

/*
     This function returns the value at the top of the stack.  Given that this operation should
     not be performed on an empty stack, the function first checks to make sure that the stack
     is not empty.  If the stack is empty, then the function throws an assertion error and the
     program terminates.  If the stack is not empty, then the function returns the value at the
     top of the stack.
*/
TYPE linkedListStackTop(struct linkedListStack *s) {
     assert(!linkedListStackIsEmpty(s));
     return s->firstLink->value;
}
/*
     This function pops (removes) the top (last) element of the stack. Given that this
     operation should not be performed on an empty stack, the function first checks to make
     sure that the stack is not empty. If the stack is empty, the function throws an assertion
     error and the program terminates. If the stack is not empty, the function updates the
     firstLink pointer to skip over the current firstLink node and point at the next node in
     the stack. Then it frees the former firstLink node, effectively removing it from the
     linked list and therefore popping it from the stack.
*/
void linkedListStackPop(struct linkedListStack *s) {
     assert(!linkedListStackIsEmpty(s));
     struct linkedListStack* temp = firstLink;
     s->firstLink = (s->firstLink)->next);
     free(temp);
}

/*
     This function determines whether or not the stack is empty.  It does so by comparing
```

```
     firstLink to 0.  If firstLInk is equal to 0, then the function returns 1; this indicates
     that the stack is empty.  Otherwise, the function returns 0; this indicates that the stack
     is not empty.
*/
int linkedListStackIsEmpty(struct linkedListStack *s) {
     return s->firstLink == 0;
}
```

Piazza Discussion

https://piazza.com/class/ib2kus4hsie528?cid=118