

# CS 232 Introduction to C and Unix

## Project 1: Crawling the Web (Due on March 2, 11:59pm)

Although you will have two weeks to work on this project, please start early. If you get stuck, please feel free to come to office hours. I am here to help you, but I cannot help if you do not ask.

***READ THE ENTIRE PROJECT DESCRIPTION CAREFULLY BEFORE BEGINNING THE PROJECT.***

### **Project goals:**

This project will deepen your knowledge of pointers and memory management by giving you hands-on experience coding these concepts. You will also gain experience working with self-referential structures and recursive functions that operate on those data structures. Additionally, this project will increase your familiarity with the basics of C (e.g., loops, basic data types, arrays, etc.) and the development process on \*nix machines.

This project is the first in an exciting sequence of projects that deal with exploring the web. These projects will build upon each other, so that in the end you will have built a simple search engine that works on real web data!

- This project involves the web crawler that jumps around to different web pages.
- The next project will involve processing the content of a web page.
- The final project will use the crawler and the content processor to build an index of web pages. That index will be used to find matches to web searches.

### **Project description:**

#### **1. Create a new Cloud9 workspace using the Python template and prepare the workspace**

Since we will use the Python program, please create a new Cloud9 workspace using the Python template, as shown in Figure 1.

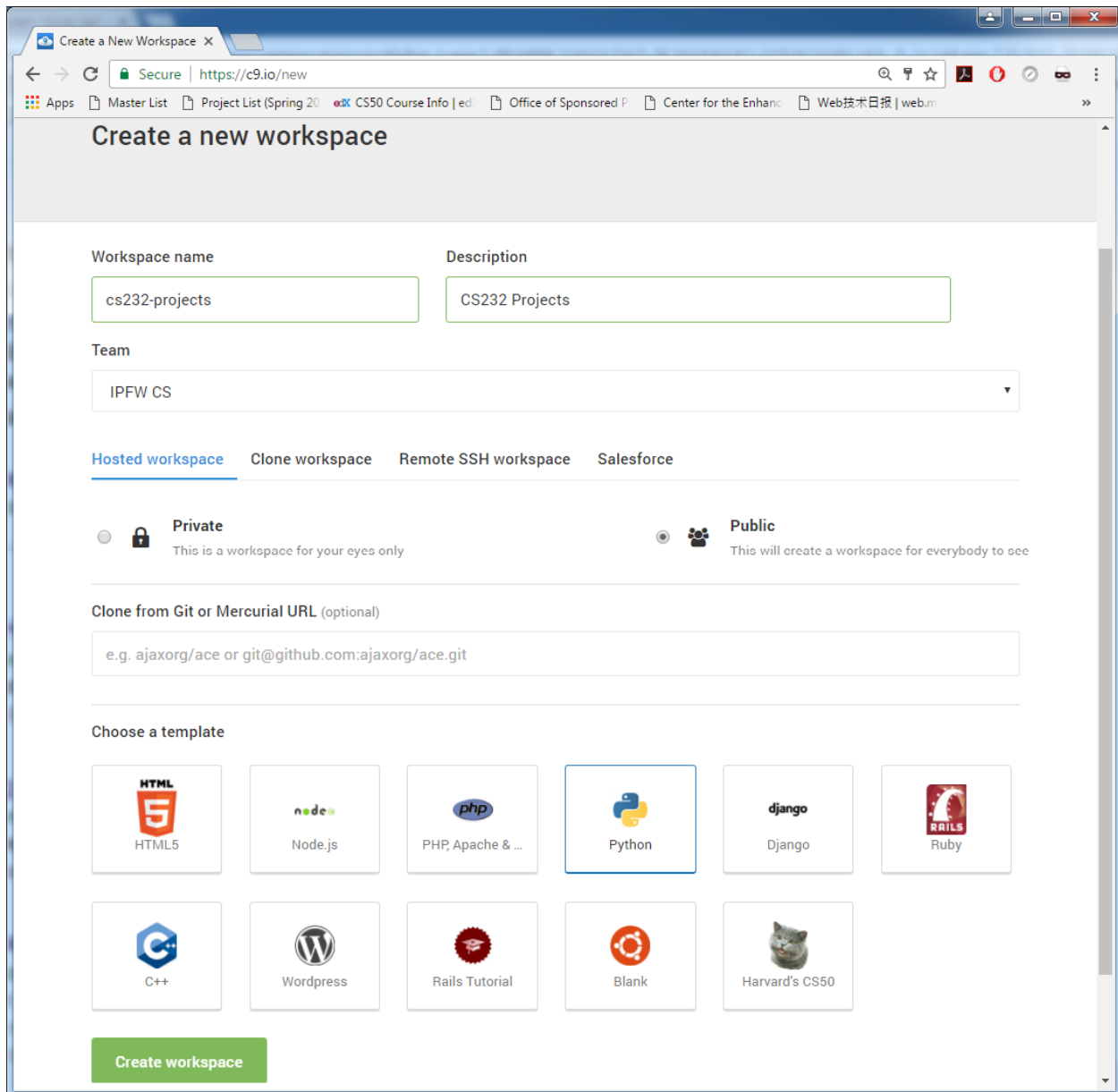
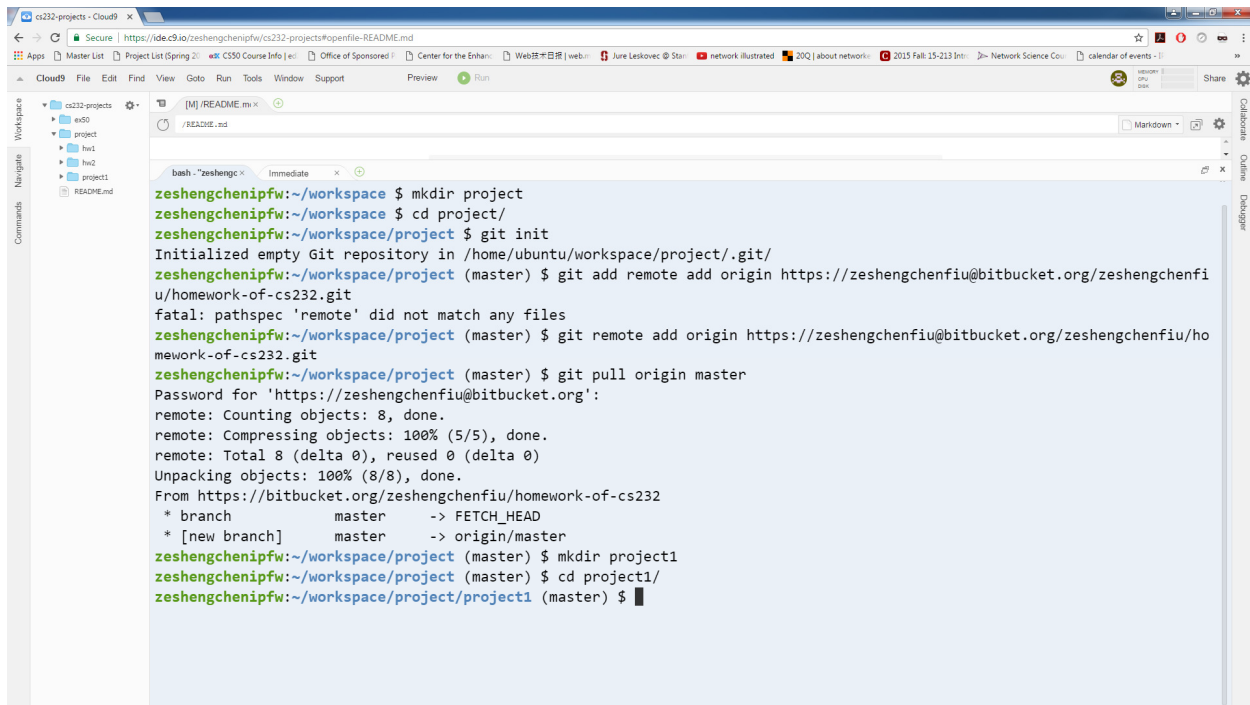


Figure 1. Create a new Python workspace in Cloud9

In the new Python workspace, please implement the following:

- Create a new directory “project”.
- Under the “project” directory, create a new git repository (i.e., “git init”).
- Under the “project” directory, pull all previous homework submissions from your Bitbucket repository (i.e., “git remote add origin your\_bitbucket\_link” and “git pull origin master”).
- Create a new subdirectory “project1” under the “project” directory.

An example is provided in Figure 2.



```
zeshengchenipfw:~/workspace $ mkdir project
zeshengchenipfw:~/workspace $ cd project/
zeshengchenipfw:~/workspace/project $ git init
Initialized empty Git repository in /home/ubuntu/workspace/project/.git/
zeshengchenipfw:~/workspace/project (master) $ git add remote add origin https://zeshengchenfui@bitbucket.org/zeshengchenfui/homework-of-cs232.git
fatal: pathspec 'remote' did not match any files
zeshengchenipfw:~/workspace/project (master) $ git remote add origin https://zeshengchenfui@bitbucket.org/zeshengchenfui/homework-of-cs232.git
zeshengchenipfw:~/workspace/project (master) $ git pull origin master
Password for 'https://zeshengchenfui@bitbucket.org':
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (8/8), done.
From https://bitbucket.org/zeshengchenfui/homework-of-cs232
 * branch          master      -> FETCH_HEAD
 * [new branch]     master      -> origin/master
zeshengchenipfw:~/workspace/project (master) $ mkdir project1
zeshengchenipfw:~/workspace/project (master) $ cd project1/
zeshengchenipfw:~/workspace/project/project1 (master) $
```

Figure 2. Sync your Python workspace with your Bitbucket repository

## Install Python bs4 module

Please download crawler.c and getLinks.py files from Blackboard, and upload them to Cloud9 Python workspace under the subdirectory “project1”.

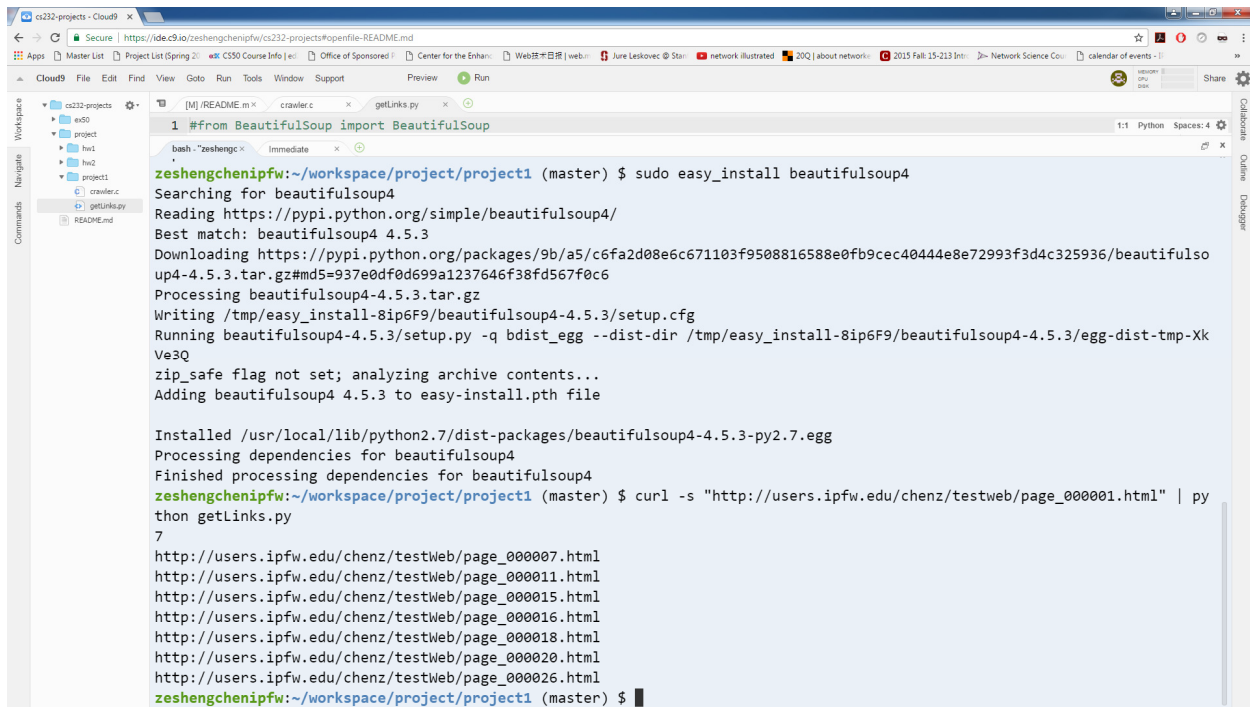
Please run

```
$ sudo easy_install beautifulsoup4
```

to install bs4 Python module. To test that the module has been installed successfully, please run

```
$ curl -s "http://users.ipfw.edu/chenz/testweb/page_000001.html"
| python getLinks.py
```

If you do not see the “ImportError: No module named bs4” error and can get the 7 URLs, it means that you have installed the Python module correctly. An example is shown in Figure 3.



```
zeshengchenipfw:~/workspace/project/project1 (master) $ sudo easy_install BeautifulSoup4
Searching for BeautifulSoup4
Reading https://pypi.python.org/simple/beautifulsoup4/
Best match: BeautifulSoup4 4.5.3
Downloading https://pypi.python.org/packages/9b/a5/c6fa2d08e6c671103f9508816588e0fb9cec40444e8e72993f3d4c325936/beautifulsoup4-4.5.3.tar.gz#md5=937e0df0d699a1237646f38fd567f0c6
Processing BeautifulSoup4-4.5.3.tar.gz
Writing /tmp/easy_install-8ip6F9/beautifulsoup4-4.5.3/setup.cfg
Running BeautifulSoup4-4.5.3/setup.py -q bdist_egg --dist-dir /tmp/easy_install-8ip6F9/beautifulsoup4-4.5.3/egg-dist-tmp-XkVe3Q
zip_safe flag not set; analyzing archive contents...
Adding BeautifulSoup4 4.5.3 to easy-install.pth file

Installed /usr/local/lib/python2.7/dist-packages/beautifulsoup4-4.5.3-py2.7.egg
Processing dependencies for BeautifulSoup4
Finished processing dependencies for BeautifulSoup4
zeshengchenipfw:~/workspace/project/project1 (master) $ curl -s "http://users.ipfw.edu/chenz/testweb/page_000001.html" | python getLinks.py
7
http://users.ipfw.edu/chenz/testWeb/page_000007.html
http://users.ipfw.edu/chenz/testWeb/page_000011.html
http://users.ipfw.edu/chenz/testWeb/page_000015.html
http://users.ipfw.edu/chenz/testWeb/page_000016.html
http://users.ipfw.edu/chenz/testWeb/page_000018.html
http://users.ipfw.edu/chenz/testWeb/page_000020.html
http://users.ipfw.edu/chenz/testWeb/page_000026.html
zeshengchenipfw:~/workspace/project/project1 (master) $
```

Figure 3. Install Python module bs4

## 2. Work on a Web Crawler

For this project, you will be completing a web crawler. The web crawler will start at some web page, follow a random link on that page, then follow a random link on that new page, and so on, keeping a list of the pages that it has visited. Your part of this project will be to complete the linked list data structure that stores the web addresses that the crawler visits. Note that this project deals with a singly-linked list (unlike the doubly-linked list we discussed in class). Singly-linked lists are simpler, because each node just points to the next node (i.e., no “previous” pointers). The last node in the list has a NULL next pointer to indicate the end. (FYI, the disadvantage with singly-linked lists is that you can’t traverse them backwards, while doubly-linked lists let you go forward and backward.)

The file `crawler.c` contains some code already written. You should only modify that file in the places that are marked “TODO: complete this”. There are comments to help guide you through the code. Note that there are four functions that operate on the linked list that you need to complete: `contains`, `insertBack`, `printAddresses`, and `destroyList`. Each of these functions must be recursive. The comments indicate how those functions should behave. Only “`printAddresses`” should print any output to the terminal.

### *Running the Program*

When completed, the program will take a web address and a number of hops from the command line and try to crawl the web for the given number of hops starting from the given

address. It will print out the addresses that it found as it was crawling, along with some messages if it found a cycle or if it ran into a dead end. For example, if you run

```
./crawler "http://www.ipfw.edu" 10
```

you might see output like

```
seed = 1486570735
Cycle detected on hop 2: address http://www.ipfw.edu
Cycle detected on hop 8: address
http://www.ipfw.edu/search/sitemap.html
Cycle detected on hop 8: address http://www.ipfw.edu
http://www.ipfw.edu
http://www.ipfw.edu/search/sitemap.html
http://library.ipfw.edu/
http://library.ipfw.edu/about/gifts-endowment.html
http://ipfw.edu/accreditation/
http://www.ipfw.edu/vcsa/
http://calendar.ipfw.edu/
http://www.ipfw.edu/
http://inside.ipfw.edu/
http://inside.ipfw.edu/tagged/iu
http://inside.ipfw.edu/tagged/ipfw
```

Because the crawler follows random links, your output will probably not match this exactly. The randomness can make debugging difficult, because running the program different times will usually lead to different execution paths. To alleviate this problem, the program prints out "seed = <some number>" at the beginning. If you add this seed as an additional command-line argument when running your program, like

```
./crawler "http://www.ipfw.edu" 10 1486570735
```

then you should get exactly the same behavior as when previously using that seed (until the web pages change!). You can do this with any seed value in order to get the same behavior multiple times.

A couple other notes that arise from dealing with real web data:

- You might get errors from the Python script "getLinks.py" that parses the web pages (because they might be malformed or in an unexpected language). If so, just rerun your program to get a different random crawling trajectory.
- Because of the network traffic involved in fetching these pages, you should limit the number of hops to 50 or fewer when you are running your project. On the due date, keep the number of hops at 10 or fewer, to allow everyone ample resources on the Cloud9 server. Thanks!

- To help check your solutions, I have created a set of web pages that link to each other and will remain the same throughout this project. Here are some example traces, which should match your program's output exactly:

- `$ ./crawler "http://users.ipfw.edu/chenz/testweb/page_000001.html" 10 899`  
seed = 899  
Cycle detected on hop 3: address `http://users.ipfw.edu/chenz/testWeb/page_000003.html`  
Dead end on hop 5: no outgoing links  
`http://users.ipfw.edu/chenz/testweb/page_000001.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000011.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000003.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000010.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000005.html`
- `$ ./crawler "http://users.ipfw.edu/chenz/testweb/page_000010.html" 5 983`  
seed = 983  
Dead end on hop 5: no outgoing links  
`http://users.ipfw.edu/chenz/testweb/page_000010.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000021.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000004.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000028.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000000.html`
- `$ ./crawler "http://users.ipfw.edu/chenz/testweb/page_000003.html" 4 187`  
seed = 187  
Cycle detected on hop 2: address `http://users.ipfw.edu/chenz/testWeb/page_000028.html`  
`http://users.ipfw.edu/chenz/testweb/page_000003.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000028.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000013.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000004.html`  
`http://users.ipfw.edu/chenz/testWeb/page_000002.html`

**To successfully complete the assignment**, you must complete the four linked-list functions in `crawler.c` so they are **recursive** functions that behave as the comments indicate. Your project should not have any memory leaks or other memory management errors. Use

```
valgrind --leak-check=yes ./crawler ...
```

to check for leaks and other memory-management errors.

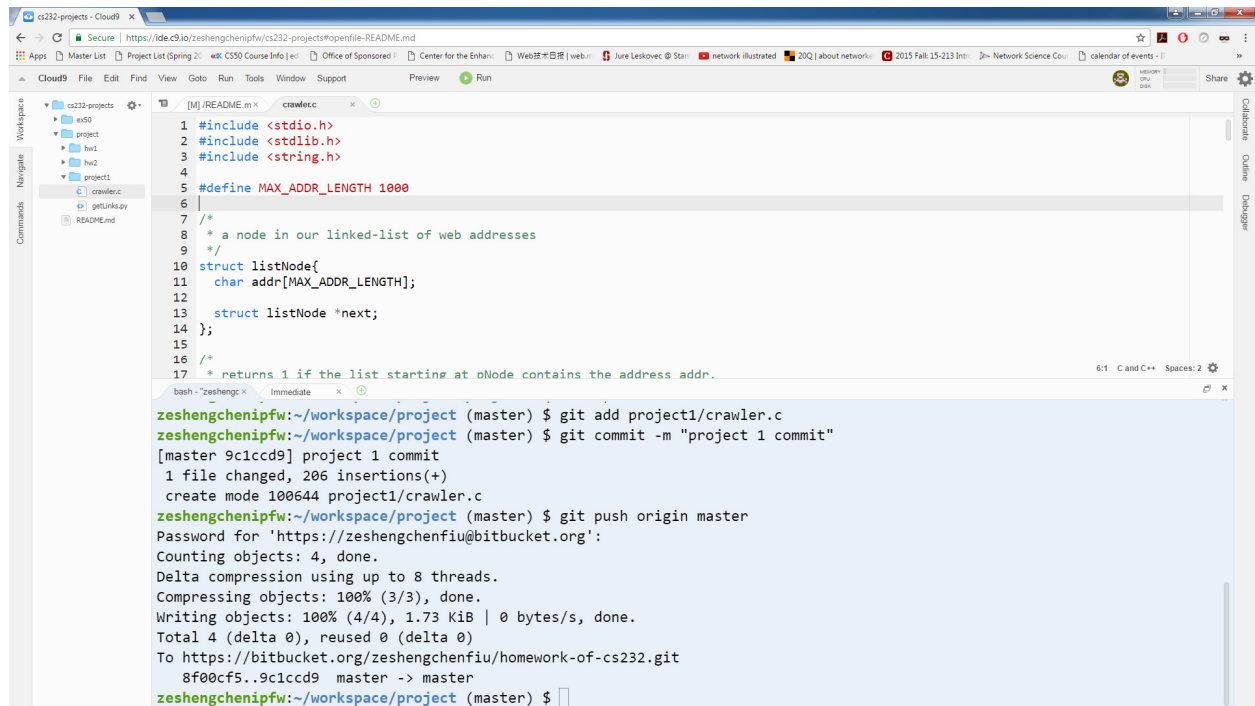
## Submission

Before pushing the file `crawler.c`, please use Git to run

```
$ git pull origin master
```

TA may have updated the score file to your Bitbucket repository. Running the above command can get the latest score file and keep your cloud9 homework directory in sync with your Bitbucket repository.

Remember that when you push the code for this project to your Bitbucket repository, please operate under the “project” directory, instead of “project1” subdirectory. An example is shown in Figure 4.



The screenshot shows the Cloud9 IDE interface. The left sidebar displays a file tree with the following structure:

- cs232-projects
  - cs232-projects
    - project
      - project1
        - crawler.c
        - getlinks.py
        - README.md

The main editor window shows the content of `crawler.c`:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define MAX_ADDR_LENGTH 1000
6
7 /*
8  * a node in our linked-list of web addresses
9  */
10 struct listNode{
11     char addr[MAX_ADDR_LENGTH];
12     struct listNode *next;
13 };
14
15 /*
16  * returns 1 if the list starting at pNode contains the address addr.
17  */
```

The terminal window at the bottom shows the following commands and output:

```
zeshengchenipfw:~/workspace/project (master) $ git add project1/crawler.c
zeshengchenipfw:~/workspace/project (master) $ git commit -m "project 1 commit"
[master 9c1ccd9] project 1 commit
1 file changed, 206 insertions(+)
create mode 100644 project1/crawler.c
zeshengchenipfw:~/workspace/project (master) $ git push origin master
Password for 'https://zeshengchenfiu@bitbucket.org':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.73 KiB | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/zeshengchenfiu/homework-of-cs232.git
8f00cf5..9c1ccd9 master -> master
zeshengchenipfw:~/workspace/project (master) $
```

Figure 4. Push the code of project 1 to your Bitbucket repository

If you have any questions, please let the instructor know.

### Grading rubric:

- Code can be compiled – 20pt
- Implement function “contains” correctly – 30pt
- Implement function “insertBack” correctly – 60pt
- Implement function “printAddresses” correctly – 30pt
- Implement function “destroyList” correctly – 30pt
- No memory leaks – 20pt
- Coding style – 10 pt

Total: 200 points.