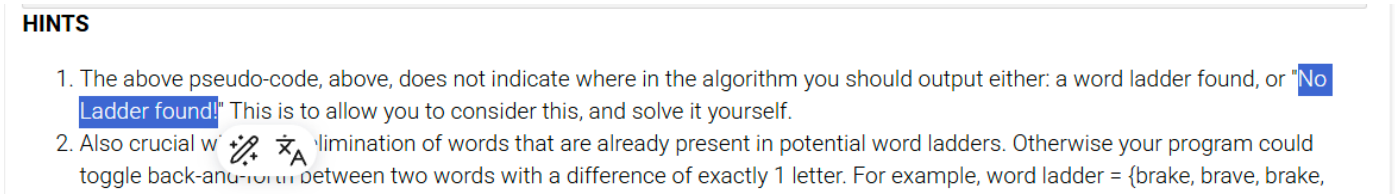
CS010C

Lab6(No.6 Lab)

Review

- Lab3 Assignment: Always the I/O
  - "No Ladder found!"
  - I just copied and pasted the description



HINTS

1. The above pseudo-code, above, does not indicate where in the algorithm you should output either: a word ladder found, or "No Ladder found!" This is to allow you to consider this, and solve it yourself.
2. Also crucial w[illegible]limination of words that are already present in potential word ladders. Otherwise your program could toggle back-and-[illegible]between two words with a difference of exactly 1 letter. For example, word ladder = {brake, brave, brake,

Lab4 Assignment

- Before you start
  - Create an Amazon Linux 2023 (not Amazon Linux 2)
    - Amazon Linux 2 have no enough space to install gdb tools
    - Already running 2023? Good to go.
  - You'll need to show coding receipts for credit in lab 4!

- Use "main_hard_code.cpp"
- BST
  - Implement "preorder", "inorder", "postorder"
- GDB
  - Set breakpoint
  - Print some information
- Graphviz
  - Implement function setting "depth"
  - Implement "write_to_file"

- Work in pairs
- Demo on Tuesday, Aug 19 (next week, No.7 Lab)
  - I'll change 1~2 nodes
  - Make sure your code still works correctly

BST

BST.H

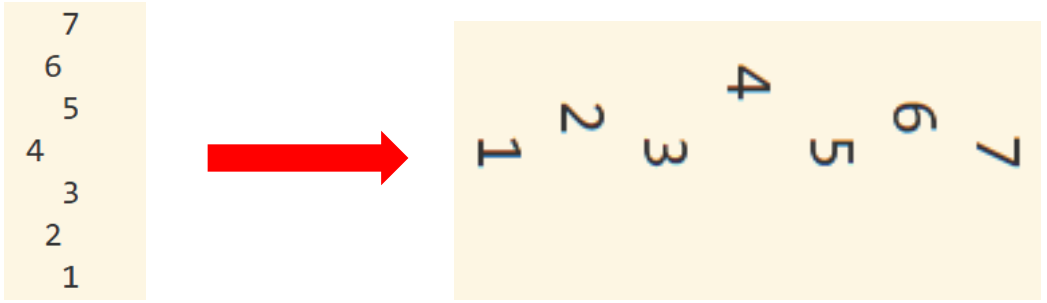
```
struct BinaryNode {

  int value; // key
  BinaryNode* left;
  BinaryNode* right;


  int depth;
  int height;
  int inorder_num;
  int preorder_num;
  int postorder_num;
```

- Completed
  - Node
    - Member variables of Node
    - insert & remove
  - BST
    - root
    - insert & remove
    - display
- You need to do
  - **preorder, inorder, postorder**
  - **write_to_file (for Graphviz)**

"static"?
Don't focus on it too much.
(simply means that every instance shares a 【single】 function/variable)

```
static BinaryNode* insert(int v, BinaryNode* t) {…
static BinaryNode* remove(int v, BinaryNode* t) {…
```

```
void display( BinaryNode* t ) {
  // in-order traversal with indented display.
  static int depth = 0;
  ++depth;
```



Demo

- Use “main-hard-code.cpp”

- Commands
  - g++ -W -Wall -Werror -g -std=c++14 main-hard-code.cpp
  - ./a.out

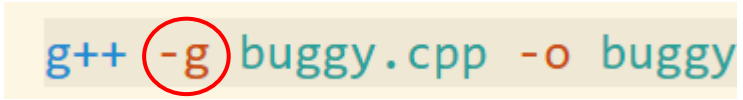
- I’ll change 1~2 nodes next week

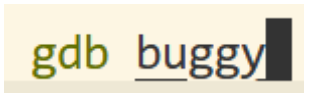
GDB

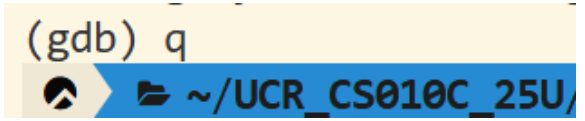
GDB: Proper Name
gdb: Executable Command

gdb

- Installation gdb for Amazon Linux 2023
  - `sudo dnf debuginfo-install glibc-2.34-52.amzn2023.0.10.x86_64 libgcc-11.4.1-2.amzn2023.0.2.x86_64 libstdc++-11.4.1-2.amzn2023.0.2.x86_64`
- Compilation
  - Add “-g”
- Load into gdb
  - “gdb your_executable”
- Quit
  - “q” or “quit”







gdb Commands

- Add a breakpoint
  - “b somewhere”
- Breakpoint information
  - “info b”
- Delete a breakpoint
  - “d breakpoint_id”
- Run
  - “r” or
  - “r arg1 arg2 ⋯” (if you need args)

```
# breakpoint
b buggy.cpp:7
b calculate_sum
```

```
# info
info b
```

```
# delete
d 2
```

```
# run
r
```

gdb Commands

- Print variable
  - “p somevar”
- Continue (till next breakpoint)
  - “c”
- Step over next line
  - “n” or
  - “n num_times”
- Step into next line
  - “s” or
  - “s num_times”

```
# print
p i
p size
```

```
# next (without calling function)
n
```

```
# step (can call function)
s
s 2
```

gdb Commands

- Backtrace
  - “bt”
  - Shows the function call stack
    - where you are and how you got here

```
# backtrace
bt
```

- ~~Bug~~(just for fun, ignore this part)
  - ~~arr[size]~~

```
for (int i = 0; i <= size; i++)
    sum += arr[i];
```

```
The sum is: 32917
It should be: 150
```

```
-----BUG HERE-----
i:$3 = 5
size:$4 = 5
arr[i]:$5 = 32767
```

Demo

1. Set breakpoint at `void preorder( BinaryNode* t )`

```
Breakpoint 1 at 0x401674:
```

2. Show breakpoint information

```
Num     Type           Disp Enb
1       breakpoint     keep y
```

3. Print the value of
   - root node,
   - its right child,
   - right child's right child

```
$1 = 4
$2 = 6
$3 = 7
```

4. Use some commands to advance to the specified node
   - Continue (recommended)
   - Next
   - Step (not recommended for this case)
5. At the leftmost node (or the minimum node)
   - Print its value
   - Print backtrace

```
$4 = 1
#0  BST::preorder (this=0x7fffffffdf38,
#1  0x00000000004016be in BST::preorder
#2  0x00000000004016be in BST::preorder
#3  0x0000000000401652 in BST::preorder
#4  0x0000000000401276 in main () at mai
```

Graphviz

Graphviz

- Implement a function setting “depth” for each node
- Implement “write_to_file”
  - Output
    - Labels: value and depth
    - edges
  - To “depth.dot” file
  - The style doesn't matter
  - The node names don't matter
    - Just suggestion
      - Char “a”, “b”, “c”, ···
      - “n”+id
- Demo
  - Run “write_to_file”, get “depth.dot” file
  - Use online Graphviz viewers
    - https://dreampuf.github.io/GraphvizOnline/?engine=dot
    - https://sketchviz.com/new
  - Show your image, including the value and depth

Both OK

```
digraph G {

  // nodes
  n4 [label="key=4, in=4"];
  n2 [label="key=2, in=2"];
  n6 [label="key=6, in=6"];
  n1 [label="key=1, in=1"];
  n3 [label="key=3, in=3"];
  n5 [label="key=5, in=5"];
  n7 [label="key=7, in=7"];

  // edges
  n4 -> n2;
  n4 -> n6;
  n2 -> n1;
  n2 -> n3;
  n6 -> n5;
  n6 -> n7;
}
```

```
digraph G {

  a[label="key=4,pre=0"]
a->b
a->c
  b[label="key=2,pre=0"]
b->d
b->e
  c[label="key=6,pre=0"]
c->f
c->g
  d[label="key=1,pre=0"]
  e[label="key=3,pre=0"]
  f[label="key=5,pre=0"]
  g[label="key=7,pre=0"]

}
```