Data Structures, Midterm Answers

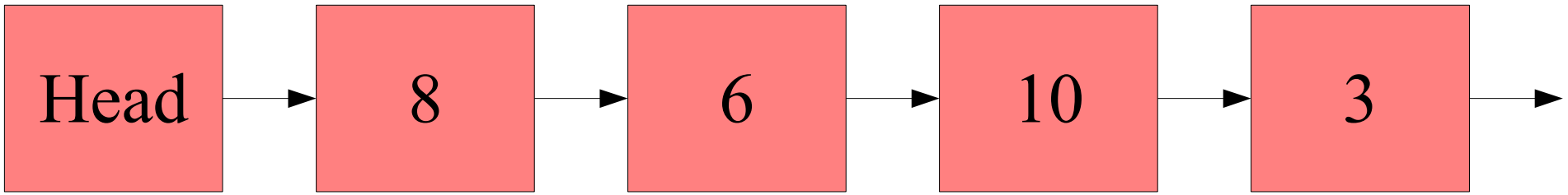
Comp-206 : Introduction to Software Systems
Lecture 14

Alexandre Denault
Computer Science
McGill University
Fall 2006

Data Structures in C

- To better understand the importance of pointers, let's take a look at two data structures.
  - Linked List
  - Binary Trees
- The source code for these examples is on the web, in the Supplemental Notes section.

Linked List



Link List Node

```
/* Node for the link list */
typedef struct ll_node {
        int value;
        struct ll_node* next;
} llnode, linkedlist;
```

Creating a node

```
/* Create a link list */
linkedlist* createLinkedList() {

        llnode* head;

        // Our implementation of linklist has a
        // dummy node at the head.
        head = (llnode *)malloc(sizeof(llnode));
        head->value = 0;
        head->next = NULL;

        return head;
}
```

Adding a Node

```
/* Add a value to the linklist. */
void addToLinkedList(linkedlist* list, int value) {

        llnode* freeSpot;
        llnode* newNode;

        // Find a free spot at the end to add the value
        freeSpot = list;
        while(freeSpot->next != NULL) {
                freeSpot = freeSpot->next;
        }

        newNode = (llnode *)malloc(sizeof(llnode));
        newNode->value = value;
        newNode->next = NULL;
        freeSpot->next = newNode;
}
```

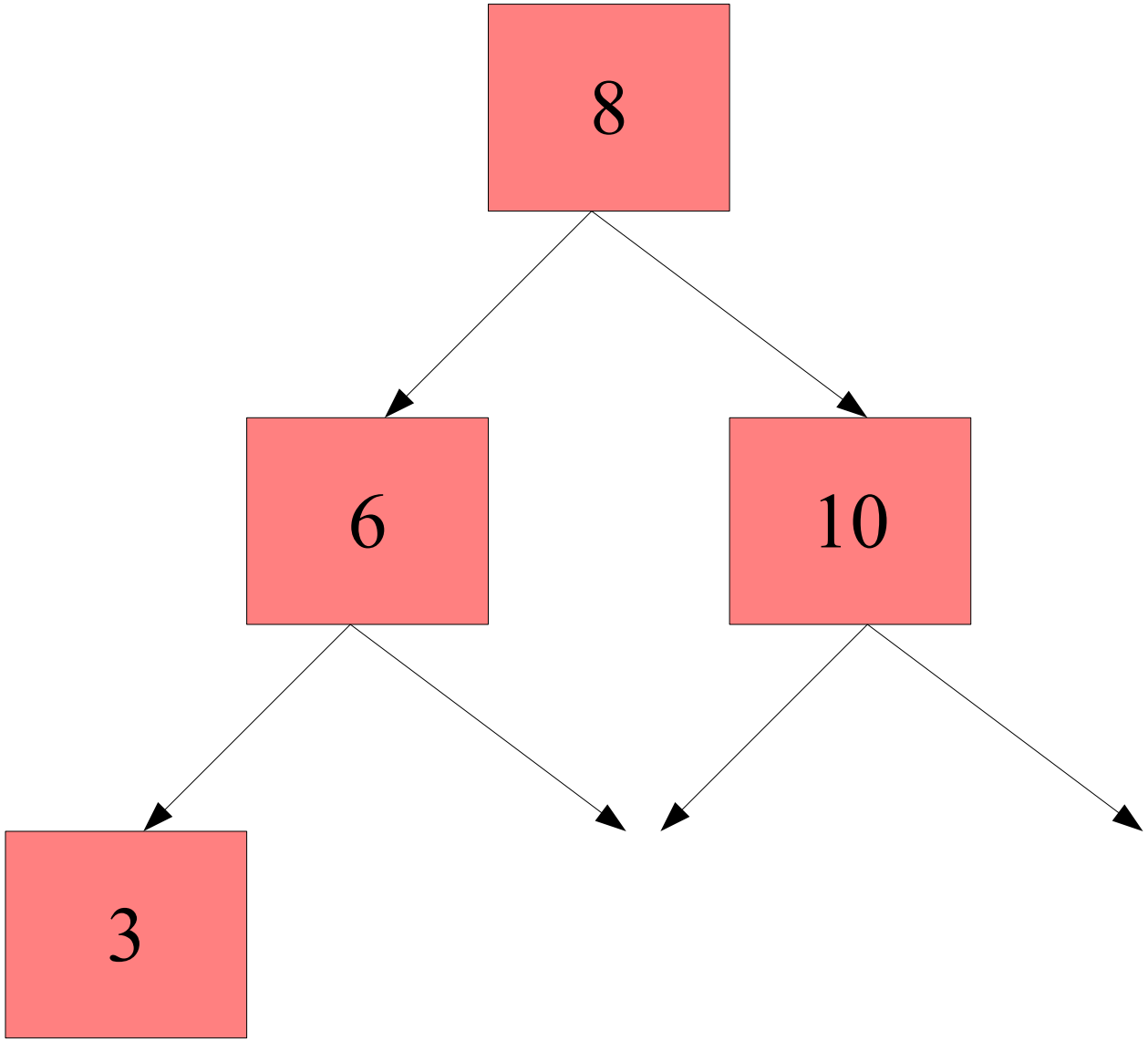
Pretty Print

```
/* Pretty print the list. */
void printLinkedList(linkedlist* list) {

        if (list->next != NULL) {
                printf("Content of list is :");
                printLLNode(list->next);
        } else {
                printf("List is empty.");
        }

        printf("\n");

}
```

Printing A Node

```
void printLLNode(llnode* my_node) {

        printf(" %i ", my_node->value);
        if (my_node->next != NULL) {
                printLLNode(my_node->next);
        }
}
```

Binary Tree



Binary Tree Node

```
/* Node in our binary tree */
typedef struct bt_node {
        char* value;
        struct bt_node* left;
        struct bt_node* right;
} btnode, binarytree;
```

Creating the Tree

```
binarytree* createBinaryTree(char* value) {

        return createBTNode(value);
}
```

Creating a Node

```
binarytree* createBTNode(char* value) {

        btnode* new_node;

        new_node = (btnode*)malloc(sizeof(btnode));
        new_node->value = (char *)malloc(strlen(value) + 1);
        new_node->left = NULL;
        new_node->right = NULL;

        strcpy(new_node->value, value);

        return new_node;
}
```

Adding a Node

```
void addToBinaryTree(binarytree* tree, char* value) {

        // In order travel of the tree, until we hit a left.
        if (strcmp(tree->value, value) > 0) {
                if (tree->left != NULL) {
                        addToBinaryTree(tree->left, value);
                } else {
                        tree->left = createBTNode(value);
                }
        } else {
                if (tree->right != NULL) {
                        addToBinaryTree(tree->right, value);
                } else {
                        tree->right = createBTNode(value);
                }
        }
}
```

Freeing the Tree

```
void unallocateBinaryTree(binarytree* tree) {

        if (tree == NULL) return;

        // First dellocate the child nodes.
        unallocateBinaryTree(tree->left);
        unallocateBinaryTree(tree->right);

        // Deallocate the node's content
        // and the node itself
        free(tree->value);
        free(tree);

}
```

Midterm

- Average 72.3%
  - 54 students took the midterm
  - 4 students got 90% or above
  - 20 students got 80% or above
  - 34 students got 70% or above
  - 44 students got 60% or above

Question 1

- Average: 7.57 / 8 : 95%
- GUI: Provides a visual interface with ready-made components (buttons, text boxes, etc / widgets) to interact with the computer.

Question 2

- Average : 3.83 / 4 : 96%
- Time Sharing / Time Slicing
  - All the processes share the CPU in turn. Each turn is called a time slice

Question 3

- Average : 3.54 / 4 : 88%
- Close to 20% of the class answered all 4 questions.
- a) An OS is closed (or proprietary) when it owned by a single company.
  - It is often designed to work on a single kind of hardware

Question 4

- Average : 3.24 / 4 : 81%
- a) /home/bob/homework/main.c
- b) ../../tmp/assignment.log

Question 5

- Average : 5.94 / 8 : 74%
- The “>” and “>>” symbols enable redirection of output on STDOUT to a file.
- If a file already exists, “>” will overwrite the file while “>>” will append the new output to it.
- The “<” enables redirection of a text file to STDIN (input).

Question 6

- Average : 13.59 / 24 : 57%
- Key elements in the answer included:
  - #!/bin/sh at the start of the script
  - Testing to see if a second argument was passed
  - Finding the user (either using $USER or whoami)
    - Note : whoami and who am i are two different commands
  - Finding the date (either using set `date` or date +%d-%m-%Y)
  - Building the proper name for the output
  - Looping over the files in the directory
  - Testing the files to see if binary (either with test or file)
    - With test, automatically get points, regardless of option
    - With file, needed to test for both “text” and ASCII
  - Properly using Tar

Question 7

- Average : 3.22 / 4 : 80%
- a) A line will match if it contains the word “Tea” or “tea”.
- b) A line will match if it starts with the letter b or a word that start with the letter b.

Question 8

- Average : 2.06 / 8 : 26%
- a) ^(([a-zA-Z]+){9}[a-zA-Z]+)$
  - Does your solution require 10 spaces to work?
  - Did you include the ^ and $
- b) [^.-][1-9][0-9][0-9]
  - Did you consider 069?
  - Did you consider 0.999?
  - Did you conisder -999?

Question 9

- Average : 4.31 / 5 : 86%
- In Python, scope is defined thought indentation.

```
if (x > 10) :
        if (x > 20) :
                print “Larger than 20”
        else
                print “Larger that 10”
```

Question 10

- Average : 2.93 / 5 : 59 %
- When coding in Python, you don't need to give variables a type.
- Types are only checked at runtime.

Question 11

- Average : 7.44 / 8 : 93 %
- Many types to choose from :
  - char : ASCII character, 1 byte
  - byte : natural number, 1 byte
  - short : natural number, 2 bytes
  - int : natural number, size depends on platform
    - → Bad example of definition : Int hold integers
  - long : natural number, 4 bytes
  - float : real number, 4 bytes
  - double : real number, 8 bytes
- Signed variable can be +/-, unsigned variable can only be +, but have a larger range.

Question 12

- Average : 8.39 / 10 : 84%
- The C Programming languages uses a single-pass compiler.
- This means variables and functions must be defined before being used.
  - Not before necessarily before main.
- Function prototyping it the declaration of those functions.
- To declare a function, simply include the signature of that function at the top of your file.

Question 13

- Average : 3.67 / 4 : 92%
- Through the malloc function.
- Because used memory is never automatically freed and you can eventually run out of memory.
  - Memory leak are a consequence of not freeing up memory.
  - By themselves, are not catastrophic.
  - However, too allocating too much memory over time without freeing it, that crashes the process.

Question 14

- Average : 2.65 / 4 : 66%
- Only the value of d was important.
  - All or nothing.
- a 6 b 8 c 6 d -1075896576