ECE 198KL Final Exam

Spring 2013

16th December, 2013

Name: _____ NetID: _____

- **This is a closed book exam except for 1 sheet of notes.**
- **You do not have to turn in this handout.**
- **You may not use a calculator.**
- **Absolutely no interaction between students is allowed.**
- **Don't panic!**

Problem number	Max	Score	Checked by
1	25		
2	25		
3	25		
4	25		
Total	100		

Problem 1: Short Answer Type Questions

Write short and precise answers to these questions in the text file **Answer1.txt**.

(a) While attempting to debug a program, Mr. Codfix decides to return the address of a local variable myvar from a subroutine called my_routine, and then to print myvar's value by dereferencing the return value from my_routine. Explain how Mr. Codfix's scheme can fail.

(b) Write a C++ class T which contains a pointer to an integer ptr. T should also have constructor(s) which initialize ptr by dynamically allocating memory for storing a single integer and initializing that integer to an integer argument passed as a parameter or to zero if no argument is given; T should also have a destructor. The constructor(s) and destructor should print the value of the integer member of the object being constructed or destructed.

(c) C++ provides two different mechanisms for dynamically allocating memory in the heap (new/malloc). Compare their relative advantages and disadvantages.

(d) Write in LC3 assembly language, the instructions that are generated by the compiler for the line `getNode = getNode->next;` you may assume that the variable getNode is stored in some named memory location OR that it is accessed using the frame pointer which is stored in register R5.

```
struct node {
  int val;
  struct node *next;
  };
int main()
{
  int value = 0;
  struct node* getNode;
  …
  getNode = getNode->next;
  …
}
```

(e) Explain **encapsulation** (scope and visibility of members) in object oriented programming with the following partial example. Which members of the class Circle are accessible from where? Does this depend on where in the program the object of the class is declared?

```
class Circle{
        float x,y;
        float radius;
 public:
        Circle(float a, float b, float r) { x = a; y = b; radius = r;}
        Circle() {x= 0; y= 0; radius = 0;}
        float Area() { …; };
        void Move(float, float);
        void Scale(float s);
};

void Circle::Move(float a, float b) {…;}

void Circle::Scale(float s) {…;}

int main( ) {
  Circle c(1,1,1);
  c.Move(10,5);
  c.Scale(5);
  cout << "Area " << c.Area() <<endl;
  return 0;
}
```

(f) The following code is supposed to insert two nodes in a linked list (with a single integer member called val and a next pointer).

(i) Explain showing the run time stack why this does not work.

(ii) Rewrite the lines with the changes (and new lines if needed) to fix the bug.

```
int addItem(iItem *head, int input) {
        iItem *newItem;
        newItem = (iItem *) malloc(sizeof(iItem));
        if(newItem == NULL) return 0;
        newItem->val = input;
        newItem->next = head;
        head = newItem;
        return 1;
}

int main() {
        int choice, input, ret;
        iItem *head = NULL;
        addItem(head, 5);
        addItem(head, 10);
        …
        return 0;
}
```

Problem 2: List Uniform

Write a program that takes a linked list and a parameter k, and produces a list such that every **run** of identical integers appearing in the input becomes a run of length k. For example, for k = 2, the linked list with 3 runs

[3] -> [3] -> [3] -> [3] -> [2] -> [2] -> [7] -| becomes [3] -> [3] -> [2] -> [2] -> [7] -> [7] -|.

Here the last two [3] nodes are deleted, the nodes with 2 are unchanged and the last node with 7 is inserted. To do this, implement the following simple functions.

(a) countRunLength. *Node* countRunLength(Node* node, int* len);*

This function traverses the list of nodes starting from 'node' until it finds the first node with a value that is different from the value of 'node'. It updates len to be the length of the run (i.e, number of nodes with value same as node), and it returns a pointer to the first node that has a different value (could be a NULL).

For example, calling **countRunLength** on the first [3] in the first list above, will have len set to 4 and return a pointer to the first [2]. On node [7], len would be set to 1 and the pointer returned would be NULL.

(b) removeNodes. *void removeNodes(Node* node, int numKeep, int numDel);*

This function takes a pointer to a node and two integers. It keeps the first numKeep nodes starting from the given node and then removes and frees the following numDel nodes with the same value.

For example, on the first [3] node in the first list with numKeep=2 and numDel=2 will make the linked list: [3] -> [3] -> [2] -> [2] -> [7] -|.

(c) insertNodes. *void insertNodes(Node* node, int numKeep, int numAdd);*

This function takes a pointer to a node and two integers. It mallocs and inserts numAdd new nodes with the same value (as node) after the first numKeep nodes starting from node.

For example, on the first [2] node in the first list above with numKeep=2, numAdd=1, gives the linked list: [3] -> [3] -> [3] -> [3] -> [2] -> [2] -> [2] -> [7] -|.

(d) changeList. *void changeList(Node* head, int k);*

This function takes a pointer to the head of the linked list and a positive integer k, to create a list with runs of length k (see the example at the beginning). It will call all the above functions with appropriate arguments. **removeNodes** and **insertNodes** should be called with the pointer to the first node in a run. For **removeNodes** numKeep + numDel = len and k = numKeep; for **insertNodes,** numKeep = len and k = numKeep + numAdd. *(continued on next page)*

Additional Details:

- Write your program in Question2/main.c
- To compile: gcc -g main.c –o main
- To run: ./main
- Couple of tests and a print function are written for you, and you may edit the main function for testing your code with different linked lists and different values of k.

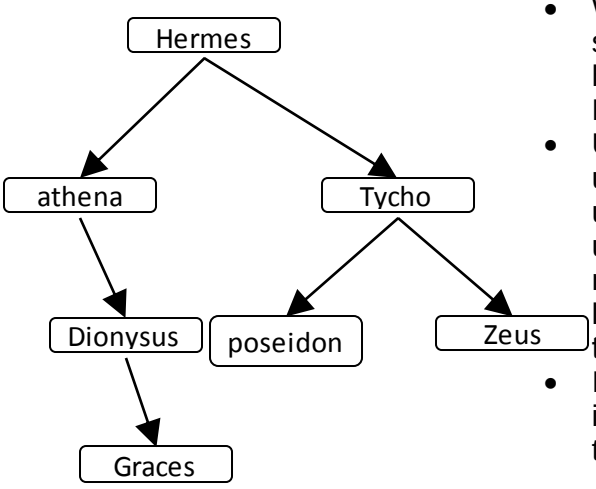
Problem 3. Binary String Tree

For this problem you will be implementing two functions for binary search tree in which the nodes store strings. The node structure for the tree is as follows

```
typedef struct node_t Node;
struct node_t {
        char *string;
        Node * left;
        Node * right;}
```

(a) **Insert:** *insert_to_bst(Node **node, char *string);*

This function inserts a node with a string as its value into the binary tree.



- When creating a node, allocate memory for storing the string. You can assume the string has a maxiumum length of 10 (9 chars + 1 NULL).
- Uppercase of the first character of the *string* is used for comparisons in insertion. If the uppercase of string[0] is less than the uppercase of the first character of the current node, then the string should be inserted in the left-subtree, and so on. (E,g, 'athena' is inserted to the left of 'Hermes').
- If a string with the same first character is already in the tree, then do not insert but print "A string that starts with %c is already in the tree\n".

(b) **Least common ancestor:** *Node* lca(Node* node, char* p, char* q):*

This function takes a pointer to a node and 2 strings p and q, and returns a node pointer to their *lowest common ancestor* or their LCA.

Definition. The lowest common ancestor or LCA of two nodes p and q is the lowest node in the given tree that has both p and q in its subtrees (a node is included in its own subtree). In the above tree LCA of Graces and Dionysus is Dionysus; LCA of poseidon and Zeus is Tycho; and LCA of Zeus and Graces is Hermes. (No, this example tree does **not** represent the actual relationship between the Olympian gods).

Use the following recursive algorithm to implement the LCA function.

- (Base case) If node is null Or node's string matches p or q, then return node
- (Recursion) Else
  - let L be the result of (recursively) calling lca on node's left child with p and q, and R be the result of (recursively) calling lca on right child with p and q.
  - If neither L nor R is null then return node
  - If L is null then return R
  - If R is null then return L
  - If both are null then return null.

Additional Details:

- Write the functions in Question3/main.c
- The gold version of insert function is given to you insertgold.o. In case you cannot figure out part (a), you can use the gold version of insertion for part (b).
- To compile: gcc -g main.c –o main
- To compile **with our insert function**: gcc -g -o main main.c insertgold.o
- To run: ./main and then choose the options for insert, LCA or display.
- A print tree function is provided for you. For the above example it will print:

```
Hermes -> athena
athena -> NULL
athena->Dionysus
Dionysus -> NULL
Dionysus -> Graces
Hermes -> Tycho
Tycho -> poseidon
Tycho -> Zeus
```

- You may use strcmp() and strlen() functions defined in **string.h**; these are describe at the end of the exam.

Problem 4c: Text redact

To **Redact** a string S of length k from a text file F means to remove every occurrence of S in F with a string xxx…x of k 'x's. For example, redacting the string "Smith" from the text "Mr. Smith was seen at the Smithsonian." will give the string "Mr. xxxxx was seen at the Smithsonian.". For this problem, you have to write a function that redacts a word from a given text file.

Write the function

```
void redact(char *inputFile, char *outputFile, char *word);
```

with three arguments (a) an input file name, (b) an output file name, and (c) a **word** (string of alphanumeric characters). This function should redact the word from the input file and print the result in the output file.

The **word** will be a string of alphanumeric characters (e.g., "smith", "5m1th"). Words are delimited by non-alphanumeric characters ("smith,adam" is 2 words). You may assume some upper bound on the length of the word to be redacted as well as all words in the input. You may use any algorithm you like; here is one suggestion:

- Read the input file one word at a time.
    - You can do this by reading the input file one character at a time and storing these characters in an array, say A, as long as the read character is alphanumeric; as soon as a non-alphanumeric character is read, A forms a word. All non-alphanumeric characters are printed to the output file.
- Compare the stored word with the **word** (to be redacted) using *strcmp* (description of strcmp provided below)
    - If the words match then for the length of the word print the character 'x' to the output file
    - Else print the word read from the input file to the output file
- As mentioned before, non-alphanumeric character should be printed to the output file.

Additional Details:

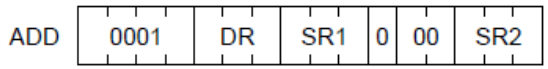
- Write your redact function in Question4/main.c
- To compile: gcc -g main.c –o main
- To test, run: ./main test.txt redacted.txt MacArthur
- Redact the words Bismarck, Archipelago, Southwest, Arawe, Peninsula to create a new file final.txt.

Two useful functions in **string.h**:

int strcmp (const char * str1, const char * str2): Returns 0 when both strings are equal.

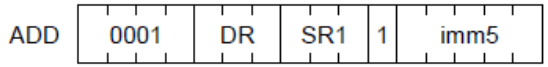
int strlen (const char * str): Returns the length of the C string str without including the terminating null character. Neither function changes the strings arguments.

NOTES: RTL corresponds to execution (after fetch!); JSRR not shown



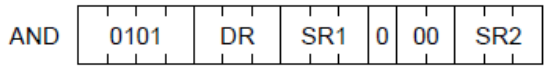
ADD DR, SR1, SR2

DR ← SR1 + SR2, Setcc



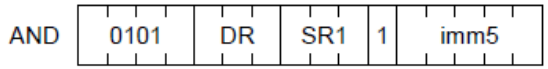
ADD DR, SR1, *imm5*

DR ← SR1 + SEXT(imm5), Setcc



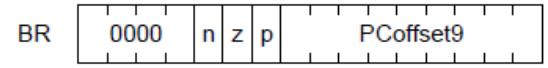
AND DR, SR1, SR2

DR ← SR1 AND SR2, Setcc



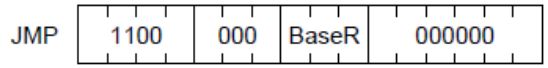
AND DR, SR1, *imm5*

DR ← SR1 AND SEXT(imm5), Setcc



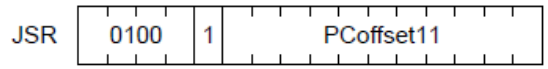
BR{nzp} *PCoffset9*

((n AND N) OR (z AND Z) OR (p AND P)):
PC ← PC + SEXT(PCoffset9)



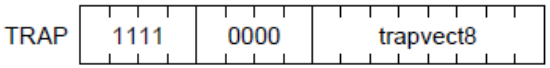
JMP BaseR

PC ← BaseR



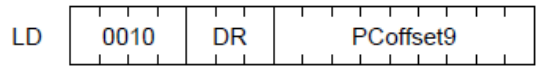
JSR *PCoffset11*

R7 ← PC, PC ← PC + SEXT(PCoffset11)



TRAP *trapvect8*

R7 ← PC, PC ← M[ZEXT(trapvect8)]

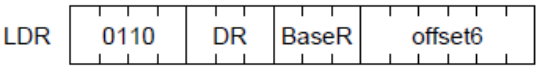


LD DR, *PCoffset9*

DR ← M[PC + SEXT(PCoffset9)], Setcc

LDI DR, *PCoffset9*

DR ← M[M[PC + SEXT(PCoffset9)]], Setcc

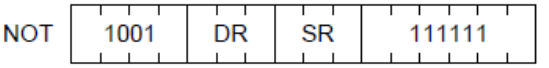


LDR DR, BaseR, *offset6*

DR ← M[BaseR + SEXT(offset6)], Setcc

LEA DR, *PCoffset9*

DR ← PC + SEXT(PCoffset9), Setcc



NOT DR, SR

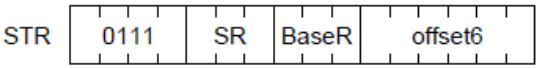
DR ← NOT SR, Setcc

ST SR, *PCoffset9*

M[PC + SEXT(PCoffset9)] ← SR

STI SR, *PCoffset9*

M[M[PC + SEXT(PCoffset9)]] ← SR



STR SR, BaseR, *offset6*

M[BaseR + SEXT(offset6)] ← SR