CSCE 2014 – Final Exam  
Spring 2014

Student Name: _____________________________

Student UAID: _____________________________

I pledge that I have neither given nor received unauthorized help on this exam.

Instructions: This is closed book exam. Students are allowed one 8.5x11 page of notes, but no calculators or other electronic devices. Please read all questions before starting the test and schedule your time accordingly.

<table>
<thead>
<tr>
<th>SECTION</th>
<th>POINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursion</td>
<td>/ 24</td>
</tr>
<tr>
<td>Linked Lists</td>
<td>/ 20</td>
</tr>
<tr>
<td>Stacks</td>
<td>/ 16</td>
</tr>
<tr>
<td>Queues</td>
<td>/ 16</td>
</tr>
<tr>
<td>Sorting</td>
<td>/ 22</td>
</tr>
<tr>
<td>Binary Trees</td>
<td>/ 20</td>
</tr>
<tr>
<td>Heaps</td>
<td>/ 16</td>
</tr>
<tr>
<td>Hash Tables</td>
<td>/ 16</td>
</tr>
<tr>
<td>TOTAL</td>
<td>/ 150</td>
</tr>
</tbody>
</table>
--- Recursion ---

Consider the following C++ code.

```cpp
int silly(int num)
{
    if (num < 0)
        return silly(-num);
    else if (num == 0)
        return -1;
    else if (num == 1)
        return 0;
    else if (num > 1)
        return (1 + silly(num/2));
}
```

(4 points) Circle all of the terminating conditions for this recursive function.

A) if (num < 0)
B) else if (num == 0)
C) else if (num == 1)
D) else if (num > 1)
E) None of the above.

(4 points) What value will the function return if we call “silly(8)” in main?

A) 8
B) -1
C) 3
D) 0
E) None of the above.

(4 points) What value will the function return if we call “silly(-4)” in main?

A) -1
B) 2
C) 1/16
D) -4
E) Infinite recursion will cause program to crash.
Consider the following C++ code.

```cpp
int funny(int num)
{
    if (num <= 2)
        return 1;
    else
        return (1 + funny(num - 1) + funny(num - 2));
}
```

(4 points) What will happen if this function is called with parameter num = 0?

A) It will return a value of 0  
B) It will cause infinite recursion and the program will crash  
C) It will return 1 + funny(-1) + funny(-2)  
D) It will return a value of 1  
E) None of the above.

(4 points) What value will the function `return` if we call `funny(4)` in main?

A) 1  
B) 3  
C) 4  
D) 5  
E) None of the above.

(4 points) Assume that you are given the standard recursive implementations of the following functions. Which function will run the **fastest** if they are executed with the same value of N?

A) `factorial(N)`.  
B) `towers_of_hanoi (N)`.  
C) `fibonacci(N)`.  
D) `binary_search(array, value, 0, N-1)`.  
E) All of the above.
Consider the following C++ code.

```cpp
class node
{
public:
    int Value;
    node* Next;
};

void fun1(node * &head, int value)
{
    node *ptr = new node();
    ptr->Value = value;
    ptr->Next = head;
    head = ptr;
}

void fun2(node* & head, int value)
{
    node *curr = head;
    node *prev = head;
    while ((curr != NULL) && (curr->Value != value))
    {
        prev = curr;
        curr = curr->Next;
    }
    if (curr != NULL)
    {
        prev->Next = curr->Next
        delete curr;
    }
}

int main()
{
    node *head = NULL;
    fun1(head, 41);
    fun1(head, 27);
    fun1(head, 56);
    // location A
    fun2(head, 27);
    fun2(head, 83);
    // location B
}
```

--- Linked Lists ---
(4 points) Draw a diagram to show what the linked list will look like at location A when we run the main program. Remember to include the head pointer.

(4 points) Draw a diagram to show what the linked list will look like at location B when we run the main program. Remember to include the head pointer.

(4 points) Assume you are given a linked list implementation that uses a head pointer but no tail pointer. Which of the following operations will be the fastest?

- A) insert_front
- B) insert_tail
- C) delete_value
- D) A and B
- E) B and C

(4 points) Assume you are given a linked list implementation that uses a head pointer but no tail pointer. Which of the following operations will be the slowest?

- A) insert_front
- B) insert_tail
- C) delete_value
- D) A and B
- E) B and C

(4 points) Assume you are given a doubly linked list implementation that uses a head pointer and a tail pointer and each node has a prev and a next pointer. Which of the following operations will be faster than a standard single linked list?

- A) insert_front
- B) insert_tail
- C) delete_value
- D) A and B
- E) B and C
---- Stacks ----

(4 points) Assume stack s contains the following integers: [4, 3, 7] where the top of the stack is on the right. Draw diagrams showing what the stack will look like after each of the sequences of instructions. Start with the initial stack in both cases.

s.push( 2 );
int num = s.top();
s.push( num+1 );

int val = s.pop();
s.push( s.pop() + s.pop() );
s.push( 8 );

(4 points) Assume you implementing a stack using an array-based approach and the variable “top” always contains the array location of the top value in the stack. Which of the following code could be used to pop “value” off the stack?

A) value = data[top]; top--;
B) top--; value = data[top];
C) value = data[top-1]; top--;
D) A or B
E) B or C

(4 points) Assume you implementing a stack using an array-based approach and the variable “top” always contains the array location of the top value in the stack. Which of the following code could be used to push “value” on the stack?

A) data[top] = value; top++;
B) top++; data[top] = value;
C) data[top+1] = value; top++;
D) B or C
E) A or C

(4 points) Which of the following would be a good application for a stack?

A) Finding the first 100 digits of PI.
B) Simulating the arrival and processing of customers at a bank.
C) Calculating the value of postfix expressions.
D) Sorting input values in ascending order.
E) None of the above.
--- Queues ---

**8 points** Assume you have a pointer-based implementation of a queue with the front of the queue at the head and the back of the queue at the tail. Draw a sequence of diagrams to show what the queue will look like after executing the following.

```
Queue q;
q.insert( 3 );
q.insert( 1 );
q.insert( 4 );

int num = q.remove();
q.insert( num );

q.insert( 7 );
q.insert( q.remove() );

int val = q.remove();
q.insert(2);
q.insert(9);
```

**4 points** Which of the following would be a good application for a queue?

A) Finding the first 100 digits of PI.
B) Sorting input values in ascending order.
C) Calculating the value of postfix expressions.
D) Simulating the arrival and processing of packets in a network.
E) None of the above.

**4 points** Circular queues can be implemented using an array-based approach where two integers are used to keep track of the locations of the front and the back of the queue. How is this approach better than a pointer-based implementation?

A) A circular queue can never become full.
B) Circular queue insertions and deletions are faster.
C) Circular queues are better for sorting data.
D) A circular queue is easier to search.
E) None of the above.
Consider the following C++ code.

```cpp
void process1(int data[], int low, int high, int &mid)
{
    // assume data values from [low .. high] are unsorted
    // choose data[low] as the pivot value
    // move values less than pivot value to locations [low .. mid-1]
    // move values greater pivot value to locations [mid+1 .. high]
    // store pivot value in data[mid] and update mid parameter
}

void sort1(int data[], int low, int high)
{
    if (low < high)
    {
        int mid = 0;
        process1(data, low, high, mid);
        sort1(data, low, mid - 1);
        sort1(data, mid + 1, high);
    }
}

void process2(int data[], int low, int high, int mid)
{
    // assume data values from [low .. mid] are sorted
    // assume data values from [mid+1 .. high] are sorted
    // merge data so values from [low .. high] are sorted
}

void sort2(int data[], int low, int high)
{
    if (low < high)
    {
        int mid = (low + high) / 2;
        sort2(data, low, mid);
        sort2(data, mid + 1, high);
        process2(data, low, high, mid);
    }
}
```
(4 points) Assume that the data array contains integer values between [0..99] and we just called "process1(data, 0, 7, mid)" from within the "sort1" function. Read the code and comments above and draw two diagrams to show what the contents of the data array would look like before and after "process1" is called.

Data before:

Data after:

Mid value:

(6 points) Use the box method to show what the sequence of recursive calls would look like if we called "sort2(data, 0, 3)" from the main program. You do NOT need to show the data values being sorted, just the sequence of recursive calls and parameter values.
(4 points) Looking at the code above, what algorithm does “sort1” implement?

A) Bubble sort
B) Selection sort
C) Merge sort
D) Quick sort
E) Heap sort

(8 points) Consider the basic implementations of sorting algorithms discussed in class and during lab. Circle true/false for each of the following statements below.

[true / false] Selection sort is always $O(N^2)$.

[true / false] Quick sort is always $O(N \log_2 N)$.

[true / false] Bubble sort $O(N)$ in the best case.

[true / false] Heap sort uses a binary search tree to sort data.

[true / false] Merge sort uses more memory than quick sort.

[true / false] Merge sort $O(N \log_2 N)$ in the best case.

[true / false] Bucket sort uses a linked list to store sorted data.

[true / false] Bucket sort is the fastest algorithm for sorting strings.
--- Binary Trees ---

(4 points) Assume we are given the following binary search tree. Modify this diagram to show what the tree would look like if we used the standard algorithm to insert the value 99 into this binary search tree.

- Root
  - 75
    - 60
    - 83
    - 43
      - 12
      - 45
    - 80
      - 81

(4 points) Assume we are given the following binary search tree. Modify this diagram to show what the tree would look like if we used the standard algorithm to delete the value 75 from this binary search tree.

- Root
  - 75
    - 50
      - 24
    - 88
      - 76
      - 91
    - 42

(4 points) Assume that you are given a full binary tree with L levels where the root node is at level 1. What is the total number of nodes that can be stored in the tree?

A) L
B) \(L^2\)
C) \(2^L - 1\)
D) \(2^{L-1} - 1\)
E) None of the above.
(8 points) Assume that we are given the following C++ declarations for a binary search tree node. Write the C++ code necessary to complete the recursive Search function that returns true if the data is found and false otherwise.

class Node
{
    public:
        int value;
        Node * left;
        Node * right;
};

bool Search(Node * node, int value)
{
    // terminating condition
    // rest of search function
}
--- Heaps ---

(4 points) Which of the following properties are NOT true for a heap?

A) The value of the left child must be less than the right child.
B) A heap must be a complete binary tree.
C) The value of a parent in the heap must be greater than all its children.
D) Insertion and deletion from a heap is \(\mathcal{O}(\log_2 N)\).
E) All of the above.

(4 points) Assume we are given the following array-based representation of a heap where the array location 0 is unused. Draw a binary tree that represents this heap.

<table>
<thead>
<tr>
<th></th>
<th>66</th>
<th>53</th>
<th>45</th>
<th>47</th>
<th>22</th>
<th>31</th>
<th>27</th>
</tr>
</thead>
</table>

(4 points) Starting with the heap array below, draw the heap after inserting the value 49 into the heap.

<table>
<thead>
<tr>
<th></th>
<th>66</th>
<th>53</th>
<th>45</th>
<th>47</th>
<th>22</th>
<th>31</th>
<th>27</th>
</tr>
</thead>
</table>

(4 points) Starting with the heap array below, draw the heap after deleting the maximum value from the heap.

<table>
<thead>
<tr>
<th></th>
<th>66</th>
<th>53</th>
<th>45</th>
<th>47</th>
<th>22</th>
<th>31</th>
<th>27</th>
</tr>
</thead>
</table>
--- Hash Tables ---

(4 points) Assume that we are given a hash table that uses “index = key % 10” and linear probing to decide where to store data. What will the hash table contain if we insert the following sequence of values {31, 41, 59, 26, 53, 58, 97, 93} in this order?

|   |   |   |   |   |   |   |   |

(4 points) Assume that we are given a hash table that uses “index = key / 10” and linear probing to decide where to store data. What will the hash table contain if we insert the following sequence of values {31, 41, 59, 26, 53, 58, 97, 93} in this order?

|   |   |   |   |   |   |   |   |

(4 points) What is the best definition of a collision in a hash table?

A) Two entries have different keys and different hash indices.
B) Two entries with different keys have the same hash index.
C) Two entries with the same key have different hash indices.
D) The hash table becomes full.
E) None of the above.

(4 points) If we have a hash table that uses linear probing and is 1000 long and 50% full, how many probes will be needed to find an open location in the hash table when we insert a new value into the table?

A) 1
B) 1.25
C) 2
D) 5
E) 50

The end!