A Previous CS 310 Final Exam

July 20, 2004

Print Your Name:

Read the following now.

- To ensure partial credit, show all your work!
- Write your name on all pages of the exam.
- Write down your answers clearly. I reserve the right to take off points due to poor writing or English structures.
- One blank page is provided at the end for your convenience.

STOP! Do not turn to the next page until instructed to do so.
1. (10pt) Show the outcome of the cout statement at the end of the following C++ code segment.

```cpp
int n=10;
int *p, *q;
int **r;
q = &n;
r = &p;
p = new int[5];
for (int i=0; i<5; i++)
    p[i] = *q - i;
p = p + 1;
*r = *r + 1;
cout << *p;
```
2. (35pt in total) Graphs.
Consider the weighted, directed graph shown below. Answer the following questions.

(a) (5pt) Argue in *one* sentence that the graph is not a DAG.

(b) (10pt) Show the adjacency list representation of the graph. (hint: don’t forget weights)
(c) (5pt) Argue in one sentence that the graph is not connected.

(d) (15pt) Using vertex 4 as the starting point, show the contents of the pred and dist arrays of the Dijkstra’s algorithm after the completion of 4 iterations.
3. (20pt) Sorting algorithms.
   Consider the array shown below. Answer the following questions.

   54, 33, 21, 25, 60, 40, 75, 83, 62, 34, 79, 90

   (a) (10pt) Show the contents of the array after applying the `partition()` routine,
   the first phase in the quicksort algorithm, to the array.

   (b) (10pt) Using the heapsort algorithm, show the contents of the array after inserting
   6 data items into the heap.
4. (25pt in total) B-trees.
Consider the B-tree shown below. Answer the following questions.

(a) (10pt) Circle the correct statement(s).
- The removal of 50 will reduce the height of the tree by 1.
- It is “possible” to add 3 data items to the tree without causing any node-splitting operation.
- It is “possible” to add 1 data item to the tree and cause the root to be split.
- It is “possible” to remove 3 data items from the tree without reducing the height of the tree.
- There is one data item in the tree whose removal will reduce the height of the tree.

(b) (15pt) Insert 45 to the tree and show the result.
5. (20pt in total) Consider the graph declaration shown below (similar to the one we used in Project 4).

```cpp
template <class T>
class Digraph {
public:
    Digraph ();
    void add_vertex (T new_vertex_label);
    void add_edge (int u, int v);
    void remove_edge (int u, int v);
    void dijkstra (int start, int* pred, int* dist);
    T operator[] (int i); // returns the label/name of vertex i
    ... other digraph methods ...

private:
    ListNode** neighbors_list;
    T* labels;
    int vertex_count;
    int max_size;
    ... you can add new private members here ...
};
```

Your task is to implement a second array reference operator

```cpp```
int operator[] (T& label);
```

so that the user of the `Digraph` class, when given the label or name of a vertex, can easily obtain the number of the vertex. To illustrate the use of the routine, consider a `Digraph<string> g` which contains 4 vertices. Presuming that vertices 0, 1, 2, 3 have names “British,” “France,” “Peru,” and “Italy,” respectively, then the expression `g["France"]` should give value 1, `g["Italy"]` should result in 3, and so on. It is mandatory that each invocation to the routine can be done in $O(\log N)$ time or less, where $N$ is the number of vertices in the graph. Furthermore, run times of the `add_vertex()` routine should also be bounded by $O(\log N)$. Answer the questions on the next page.
(a) (10pt) Discuss in *four* or less sentences how you are going to implement the routine. Note that for this question C++ coding is unnecessary, not welcome actually. Just clearly present your “idea.”

(b) (10pt) List the additional, private data members you need to add to class *Digraph* in order to implement the routine. For each item you add, explain its use/purpose in *one* sentence.
This page is intentionally left blank.