

**CS140b Spring 96 – Final exam**  
Prof. Frank Vahid

Name: \_\_\_\_\_ Student number: \_\_\_\_\_

As always, justifying your answer may help with partial credit.

1. (5 pt.) Write a C++ function, *IntsSum*, that has three integer array parameters *s1*, *s2*, and *result*, and an integer parameter *size*, and makes each *result* item the sum of the corresponding items in *s1* and *s2*. For example, if *s1* is (1 2 3 4) and *s2* is (2 3 4 5), then *result* would be (3 5 7 9) (Note: up to -10 points if you get this badly wrong or leave it blank – anyone passing CS140 should be able to answer this easily).
  
2. (3 pt.) Circle the problems that lend themselves to being solved by a graph (+1 right, -1 wrong):
  - (a) Finding the lowest-cost train routes between pairs of cities.
  - (b) Sorting a set of integers.
  - (c) Analyzing a computer network for nodes whose failure would cut-off communication between other nodes.
  - (d) Filling a knapsack to capacity while maximizing the sack's contents' value.
  - (e) Simulating the annealing process.
  - (f) Representing an arithmetic computation so we can schedule the arithmetic operations onto arithmetic units.
  - (g) Scheduling multiple processes onto a processor.
  
3. (6 pt.) Precisely define the following graph terms:
  - (a) Connected graph:
  - (b) Cycle:
  - (c) Tree:
  - (d) Vertex degree:
  - (e) Digraph:
  - (f) Connected component:
  - (g) Articulation point:

4. (5 pt.) Answer the following queries about graphs. Assume there are  $N$  vertices.
- (a) The maximum edges in a graph: \_\_\_\_\_
  - (b) The minimum edges in a graph: \_\_\_\_\_
  - (c) The longest possible simple path in a digraph: \_\_\_\_\_
  - (d) The maximum edges in a tree: \_\_\_\_\_
  - (e) The average vertex degree in a graph with  $E$  edges: \_\_\_\_\_
5. (2 pt.) List the data fields of a graph class and vertex class for an adjacency list representation, assuming vertices are represented as integers.
6. (4 pt.) Compare the adjacency matrix and adjacency list graph representations for the following:
- (a) Space complexity:
  - (b) Space efficiency for complete graphs with weighted edges:
  - (c) Time complexity of finding a vertex's adjacent vertices:
  - (d) Time complexity of determining if two vertices are adjacent:
7. (4 pt.) Write a function *Graph.DFS* for performing a depth first search on a connected graph, printing each vertex as it is visited. You may use reasonable pseudo-code, but stay as close to C++ as possible. Assume basic graph member functions have already been implemented. What's the complexity of DFS?

8. (3 pt.) Complete the following function so that it counts the number of components in a graph. Assume other graph member functions are available.

```
int Graph::CountComponents()
{
    visited = new Boolean[n];
    for (int i = 0; i < n; i++) visited[i] = FALSE;
                                     // add a statement here
    for (i = 0; i < n; i++)
        if (! visited[i]) {
            cout << endl << "New Component :";
                                     // add a statement here
                                     // add a statement here
        }
    delete [] visited;
    return count;
}
```