CSC 112 MIDTERM EXAM

You will have 110 minutes to complete this exam. All work should be written in the exam booklet. Start with the questions that you know how to do, and try not to spend too long on any one question. Partial credit will be granted where appropriate. Good luck!

1. (10 points) Predict the output generated by each of the following programs.

a.
```
#include <iostream>
using namespace std;

void main() {
     int i;
     int arr[6] = {4, -2, 6, 3, 1, -8};
     int min = arr[0];

     for (i = 0; i < 6; i++) {
          if (arr[i] < min) {
               min = arr[i];
          }
          cout << min << endl;
     }
}
```

b.
```
#include <iostream>
using namespace std;

int myfun(int a, int &b, int *c) {
     a++;
     b++;
     (*c)++;
     cout << a << endl;
     cout << b << endl;
     cout << (*c) << endl;
     return a+b+(*c);
}

void main() {
     int x = 1;
     int y = 2;
     int z = 3;

     z = myfun(x, y, &x);
     cout << x << endl;
     cout << y << endl;
     cout << z << endl;
}
```

2. (10 points) Write code to perform any necessary memory allocations and deallocations, based upon the usage of the variables below. Caution: Allocate the minimum memory needed, based upon the variable declarations and usage.

```
void main() {
     int i, j;
     int *ptr;
     int arr1d[7];
     int **arr2d;

     // put allocations here...

     // using ptr:
     *ptr = 5;

     // using arr1d:
     for (i = 0; i < 7; i++) {
          arr1d[i] = i;
     }

     // using arr2d:
     for for (i = 0; i < 6; i++) {
          for (j = 0; j < 4; j++) {
               arr2d[i][j] = 0;
          }
     }

     // put deallocations here...
}
```

3. (10 points) When performing computations on large input data sets, big-O notation gives an indication of the relative speeds of different algorithms. Rank the following programs in order from fastest to slowest according to their big-O asymptotic complexity.

a. Selection sort: quadratic

b. Linked list random access: O(n)

c. Linked list insertion: O(1)

d. Boolean satisfiability: exponential

e. Quicksort: O(n log n)

4. Suppose you are consulting on a climate study project that will gather environmental data for a collection of study plots in a nature preserve. You are designing a new class, `ClimateData`, that will keep track of rainfall for a single plot over time. Each day, volunteers will call an `update()` method, supplying the amount of rain that fell during that day. Your class should record the `daysElapsed` and `totalRainfall` since the start of the study. It need not keep track of the daily values explicitly. However, your class should provide accessors to read the number of days elapsed, total rainfall, and average rainfall per day. In your answers to this question, you should show that you have absorbed the principles of good class design taught in class.

a. (5 points) Write a class definition for `ClimateData`. Make sure that you use appropriate types for each of the properties, and use the `public` and `private` keywords appropriately. You do not need to include definitions for any of the methods yet – just the declaration within the class definition is sufficient. Your class should include all applicable accessors and manipulators, plus constructors and destructors as appropriate.

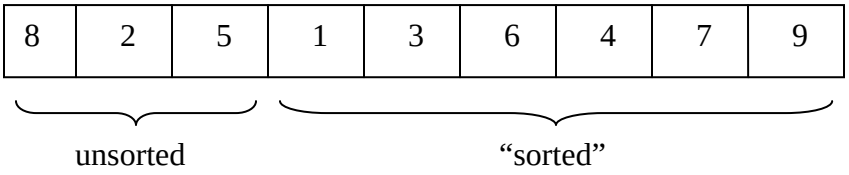
b. (5 points) Write the definition for the manipulator `update()`.

c. (5 points) Write the definition for the accessor `getAverageRainfall()`.

d. (5 points) Write the definitions of the default and copy constructors for the new class. (Make sure that the default constructor initializes values appropriately.)

5. (10 points) Write 2 to 3 paragraphs on the topic of **modularity.** Explain how C++ is set up to promote modularity, and how you can increase the modularity of your programs via good design. Be specific in your examples.

6. (10 points) Adele is implementing the insertion sort algorithm on arrays as we did on the second homework assignment. The diagram below shows the status of the array midway through the execution of her program. Due to a bug in the program, the "sorted" portion of the array is not correctly sorted. Nevertheless, your job is to simulate the completion of the insertion sort algorithm in the configuration shown, assuming that the remainder of the execution follows the algorithm without errors. Please draw the final configuration of the array as the algorithm completes.



7. (10 points) Find the bugs in the following implementation of selection sort, which is intended to sort the array in ascending order starting from the front of the array. Please mark clearly the changes you are making. (Note: assume that the comments correctly show the intended behavior.)

```
void swap(int a, int b) {
     int tmp;

     tmp = a;
     a = b;
     b = tmp;
}

void selection_sort(int *arr, int arr_len) {
     int i, j;
     int large;

     // loop through entire list
     for (i = 0; i <= arr_len; i++) {
          // find location of largest remaining item
          for (j = i; j < arrlen; j++) {
               if (arr[j] > arr[large]) {
                    large = arr[j];
               }
          }

          // swap items
          swap(arr[i],arr[j]);
     }
}
```

8. (10 points) Draw the memory structures that would be created by the following program. Make sure that you show the data values in the list nodes, as well as all the pointers.

```
#include "DL_IntList.h"

void main() {
     DL_IntList ls;
     DL_IntListNode item;

     ls.insertAtHead(5);
     ls.insertAtHead(11);
     ls.insertAtHead(6);
     ls.deleteFromTail();
     ls.insertAtTail(8);
     ls.insertAtHead(-3);
     item = ls.getHead()->getNext;
     item->insertPrev(0);
     item->excise();
}
```

9. (10 points) Draw the memory structures that would be created by the following program, and explain why it is problematic.

```
#include "DL_IntList.h"

void main() {
     DL_IntList ls;

     ls.insertAtHead(5);
     ls.getHead()->insertPrev(0);
}
```