CS 115 Exam 3, Fall 2009

Your name: _____

Rules

- You must briefly explain your answers to receive partial credit.
- When a snippet of code is given to you, you can assume
  - that the code is enclosed within some function, even if no function definition is shown
  - that the `main` function is properly defined
  - that the `iostream`, `algorithm`, `fstream`, `iomanip`, `string`, `cstring`, and `cmath` libraries have been included at the beginning of the program.
- When you are asked to write *a snippet* of code, you may assume
  - that your code is enclosed within some function
  - that any necessary libraries have been included.
- When you are asked to write *a complete program*, you must write the `#include` statements, the `int main()`, etc. in your solution to receive full credit.
- A line consisting solely of "..." represents one or more unspecified C++ statements, some of which may change the values of program variables.

Grade (instructor use only)

	Your Score	Max
Problem 1		15
Problem 2		15
Problem 3		15
Problem 4		15
Problem 5		25
Problem 6		15
Total		

Problem 1: 15 points.

Match the following descriptions with the term they describe by writing that term in the space provided. The choices of terms are:

- this
- constructor
- destructor
- class
- object
- method
- pointer
- overloading
- operator
- bubble sort
- selection sort
- linear search
- binary search
- reference
- dereference

Not all terms will be used.

(a) An algorithm that makes *N* passes over an *N*-element array, swapping adjacent elements if they are out of order

(b) A function that is part of a class

(c) The operation of following a pointer to the data it points to

(d) A variable containing the address of another variable

(e) A user-defined data type that may contain variables and functions

Problem 2: 15 points.

Using the selection sort code on the next page, which works identically to the code from your labs, show the contents of the following 5-element array after each iteration of `SelectionSort`'s loop in the labeled boxes. *Leave the boxes blank if the loop does not complete a given iteration.*

INITIAL VALUE

C	D	E	F	A

AFTER i=0

AFTER i=1

AFTER i=2

AFTER i=3

AFTER i=4

AFTER i=5

AFTER i=6

Code for Problem 2.

You may tear this page out of your exam.

```
void SelectionSort(char arr[], int size) {
   int min_pos = 0;

   // This is the loop in question
   for (int i = 0; i < size-1; i++) {
      min_pos = FindMinPos(arr, i, size);
      if (min_pos != i) {
          swap ( arr[i], arr[min_pos] );
      }
   }
}

int  FindMinPos(char arr[], int start, int size) {

   char minVal = arr[start];
   int minPos = start;

   for (int i = start+1; i < size; i++) {
     if (arr[i] < minVal) {
          minVal = arr[i];
          minPos = i;
      }
   }
   return minPos;
}

void Swap (char& a, char& b) {
   char temp = a;
   a = b;
   b = temp;
}
```

Problem 3: 15 points.

Answer the following questions about performing binary search on the array below. The binary search code from your labs is on the next page for your reference.

ARRAY CONTENTS

-5	-2	1	4	7

(a) If we are searching for the value -5, which array elements will we compare to -5 during our search?

(b) If we are searching for the value 6, which array elements will we compare to 0 during our search?

(c) What is the minimum possible number of comparisons that we could make in a binary search of this array?

(d) What is the maximum possible number of comparisons that we could make in a binary search of this array?

Code for Problem 3.

You may tear this page out of your exam.

```
/* Binary search function
  * arr = array of integers to search
  * size = size of array
  * searchInt = int to look for in the array
  *
  * Returns subscript of searchInt in array or -1 if not
found
  */
int BinarySearch(int arr[], int size, int searchInt) {
     int first_index = 0;
     int last_index = size-1;
     int middle;

     while ( first <= last ) {
          middle = (first + last) / 2;
          if ( searchStr == arr[middle] ) {
               return middle;
          }
          else if ( searchStr < arr[middle] ) {
               last = middle - 1;
          }
          else {
               first = middle + 1;
          }
     }
     return -1;
}
```

Problem 4: 15 points.

For this problem, you must write a **class definition** for a class named **`Pixel`** that contains the elements listed below.

Note that data members should be private and member functions should be public.

For now, you are only writing prototypes for the member functions. You will define the functions in the next problem.

- A name for the pixel (as an array of 1024 characters)
- Red, blue, and green components of the pixel (as 3 integers)
- Prototype for a default constructor
- An overloaded > operator for the pixel:
  `bool operator > (const Pixel& other) const;`
- Prototype for a function called `SetValues`. This function will take three variables of type `int` as inputs and will return a `bool`.
- Prototype for a function called `SetName`. This function will take a (`char *`) as input and will not return anything.
- Prototype for a function called `IsGrayscale`. This function will return `true` if the pixel is grayscale and `false` otherwise.
- Prototype for a function called `Invert`. This function will invert the colors of the pixel. It will not return anything.

Problem 5: 25 points.

In this problem, you will write definitions for the functions in the class `Pixel`. Here is a little bit more information about the functions. *Note that none of your code for this problem should include* `cin` *or* `cout` *statements!*

- The default constructor will initialize the name to the empty string and all of the color fields to 0.
- The > operator should return true if the *sum* of the pixel's red, green, and blue fields is greater than the sum of the other pixel's fields.
- The `SetValues` function will work as follows:
    - If one or more of the inputs is less than zero or greater than 255, it will return `false`.
    - Otherwise, it will set the red field equal to the first input, the green field equal to the second input, and the blue field equal to the third input.
- The `SetName` function will use `strcpy` to copy the input string into the pixel's name field
- `IsGrayscale` should return true if the pixel is grayscale (that is, if the values of all of the color fields are equal) and false otherwise.
- The `Invert` function should invert the colors of the pixel.

Problem 6: 15 points.

Assume that the class definitions you wrote in Problems 4 and 5 are located in a file called `pixel.h` in the same directory as the program you're about to write.

Write **a complete program** that uses the `Pixel` class from `pixel.h` to do the following:

- Create a `Pixel` object. Set its name to "pix1" and its red, green, and blue values all equal to 255.
- Create another `Pixel` object. Ask the user to supply the red, green, and blue fields, and set the fields of that `Pixel` accordingly.
- Using the member function IsGrayscale, print "Grayscale!" if the second pixel is grayscale.
- Invert the colors of both pixels.
- Print "pix1" if the inverted version of the first pixel is greater than the inverted version of the second pixel (according to the greater-than operator).

REFERENCE

C-string functions:

`strlen`	Input is a C-string. Returns the length of the string (not including the null terminator). Usage example: `length = strlen(name);`
`strcat`	Input is two C-strings. Appends the second string to the end of the first string (the first string is changed, but the second is not). Usage example: `strcat(str1, str2);`
`strcpy`	Input is two C-strings. Copies the second string to the first string, overwriting the original contents. Usage example: `strcpy(str1, str2);`
`strcmp`	Input is two C-strings. Returns 0 if they are the same, a negative number if `str2` is alphabetically greater than `str1`, and a positive number if `str1` is greater than `str2`. Usage example: `if(strcmp(str1, str2) > 0)`