CPSC 131 Homework 2

Deadline: Monday, Feb 11th (MoWe sections), Tuesday, Feb 12th (TuTh sections)
Turn in your submission as hard copy in class. Do not upload on TITANium.

#1

The following class keeps track of how many items are on a restaurant's menu, and the number of calories of each menu item. The calories of the menu items are positive values of type `unsigned int` and are stored as a dynamically allocated array. The first data member is a pointer that will point to the first element of the array. This array will be of an arbitrary size. The default size of the array shall be 25, but may be specified by the user at the time of construction. The second data member represents the size of the array, i.e. the number of menu items, and is stored as `size_t`.

```
class MenuCalories {
 private:
   unsigned int* calories;
   size_t num;

 public:
   // constructors (default, one arg, and copy)
   MenuCalories();
   MenuCalories( unsigned int numberOfMenuItems);
   MenuCalories( const MenuCalories& original);

   // destructor
   ~MenuCalories();

   // Member function
   int calorieAtIndex( int index );
};
```

i) If this is in the Class Specification (header) file MenuCalories.hpp, write the function definitions that would be in the Class Implementation (source) file MenuCalories.cpp.

ii) Why <u>must</u> the parameter of the copy constructor be of type constant reference?

#2

Define an ordinary function called DisplayMin that has three parameters: the first two parameters are parallel arrays of different types and the third argument is of type size_t representing the size of both arrays. The function DisplayMin must work when called with various types of actual

arguments, for example
string DisplayMin(string names[], int calories[], int size) or
int DisplayMin(int calories[], float prices[], int size).
(Parallel Arrays are two arrays whose data is related by a common index. For example: with the string array and int array, index 3 of the string array would have some name, and index 3 of the int array would have an associated value to that name (say, number of calories).)

Write **one** function that will take either set of arguments in, displays and return the value from the first array corresponding to the min value from the second array, in the form "The min value of <int/float value> belongs to <string/int value>."

Test your function by calling it with different inputs from a main() function and then printing the result of the function in the main program.

Examples:

Given:

names => string[] = {"French Fries", "Cheeseburger", "Hamburger", "Soda", "Coffee"}
calories => int[] = {300, 650, 550, 450, 50}
size => int = 5

Output:

"The min value of 50 belongs to Coffee."

Given:

calories => int[] = {300, 35, 650, 550, 450, 50}
prices => float[] = {2.99, .99, 2.79, 3.19, 1.79, 2.15}
size => int = 6

Output:

"The min value of 35 belongs to 0.99."

#3

a) In general, when is it better to use **call-by-value** over call-by-reference for a function argument? Give an example of a programming scenario (aka a function) where is certainly better to use call-by-value for one of the parameters.
b) Now the opposite question: in general, when is it better to use **call-by-reference** over call-by-value for a function argument? Give an example of a programming scenario (aka a function) where it is certainly better to use a call-by-reference parameter.
c) Consider the following programming scenario:

```
int i = -4, j = 10;
int *ptr1 = & i;
int *ptr2 = & j;
int &ref1 = i;
int &ref2 = j;
*ptr2 = *ptr1;
cout << *ptr2 << endl;
```

```
ptr2 = ptr1;
cout << *ptr2 << endl;
ref2 = ref1;
cout << ref2 << endl;
```

Circle all the instruction(s) that change the value of variable `j` after being initialized.