CSCE 310 Data Structures & Algorithms

Fall 2004
Steve Goddard

Homework 0, August 24
(Total of 100 points)

Evaluation of prerequisite knowledge

Due: 9:00pm Thursday, September 2

---

(1) 60 points: This is a "simple" C++ programming problem to help familiarize you with the Linux development environment and to refresh your programming skills. The requirements are simple:

a. Your C++ executable program will be named `PrintWithTabs`.
b. It will accept one input parameter on the command line, which will be the name of the ASCII input file. That is, your program will be execute with the command:

```
PrintWithTabs inputFile
```

c. The program will read from the input file, convert all commas to tab characters, delete all semicolons, and print the modified contents to standard output (stdout). For example, a sample line from an inputFile which reads

```
Line1,of,input,data;
Line2,forecast,input,data;
Line3,input,,data;
Line4,,,data;
```

will be output as

```
Line1   of      input   data
Line2   forecast        input   data
Line3           input   data
Line4                   data
```

d. The coding standard must be followed.
e. Create and turn in a Makefile that will build your executable from your source file(s).

You may download a test input file from the course Web page. Its URL is
http://cse.unl.edu/~goddard/Courses/CSCE310J/Assignments/ForecastDataWithNames.csv

The problem will be scored at follows:

Program correctness	30%	(18 pts)
Quality of design/readability	25%	(15 pts)
In-line documentation/coding standard	25%	(15 pts)
Design and analysis document	15%	(9 pts)
Thoroughness of test cases	05%	(3 pts)

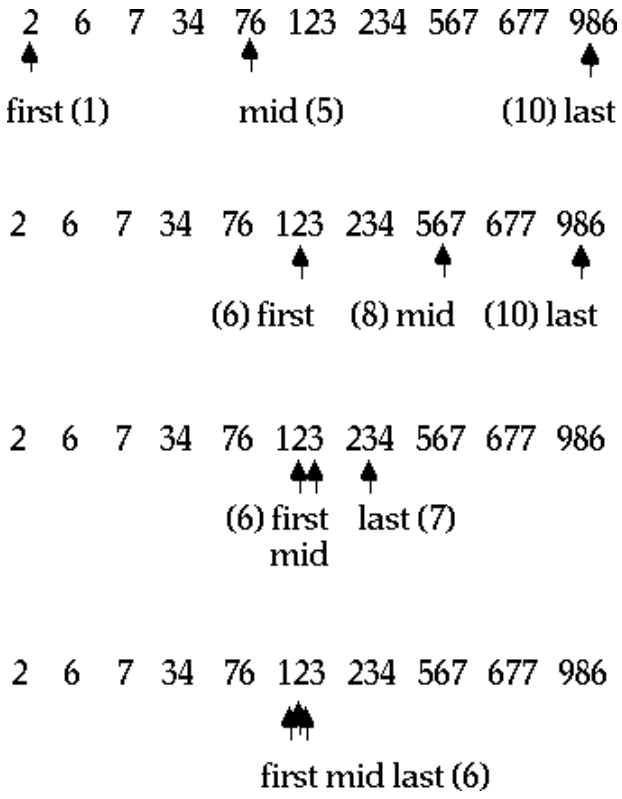
The design and analysis document should describe how your program works, what design choices (if any) were made and why, and any known defects. You may also discuss enhancements you think should be made, but were beyond the scope of the assignment (i.e., in violation of the specifications).

(2) 20 points:The following rather inelegant code is an attempt at the implementation of a binary search of an array.

"A brief description of a binary search is as follows: The binary search algorithm can only be applied if the data are sorted. You can exploit the knowledge that they are sorted to speed up the search. The idea is analogous to the way people look up an entry in a dictionary or telephone book. You don't start at page 1 and read every entry! Instead, you turn to a page somewhere about where you expect the item to be. If you are lucky you find the item straight away. If not, you know which part of the book will contain the item (if it is there), and repeat the process with just that part of the book. If you always split the data in half and check the middle item, you halve the number of remaining items to check each time. This is much better than linear search, where each unsuccessful comparison eliminates just one item."*

Binary search - Example

Seek the value 123:



*** The explanation and diagram above where taken from the cs.adfa web page.**

In the code printed below, there are four calls to binarySearch from main.

(a) What output is generated after each call to binarySearch
(b) Are these outputs expected (meaning are they what you would consider appropriate for a binary search of an array). Explain your answer.
(c) How can this code be improved. If a change in code is suggested, make the improvements and submit new code.
(d) What is the purpose of the variable n and is it necessary? Explain your answer.

```
#include <iostream>
using namespace std;
void binarySearch (int a[], int item)
{
int n =7;
int first = 0, mid, last = n-1;
bool found = false;
while (first <= last && !found){
mid = (first + last)/2;
if (item< a[mid])
{
last = mid;
}
else if (item > a[mid]){
first = mid;
}
else{
found = true;
cout << "\nfound = ";
cout << found;
cout << "\n";
}}}
void main(){
int nums[8]={25,44,62,63,84,87,92};
int value = 63;
cout<<"\nValue = ";
cout<<value;
binarySearch(nums, value);
value = 45;
cout<<"\nValue = ";
cout<<value;
binarySearch(nums, value);
value = 96;
cout<<"\nValue = ";
cout<<value;
binarySearch(nums, value);
value = 8;
cout<<"\nValue = ";
cout<<value;
binarySearch(nums, value);
}
```

(3) 20 points The following implementation is working code. However, the poor choice of variable names and the complete lack of comments as well as the nondescript prompts have rendered this code fairly unreadable.

a) Describe, in detail, what this code does.
b) Rename the variables to more intuitive names.
c) Change the words in the messages to properly prompt for what is needed.
d) Add comments to briefly explain what is occuring.
e) Submit working code with all modifications made.

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

const int FOO = 3;
typedef int BAR[FOO][FOO];
bool C(BAR b, int n, int d);
void main(){
 BAR b;
 cout << "Message 1    " << FOO << " X " << FOO << " Message 1 continued:\n";
 for (int i = 0; i < FOO; i++)    for (int j = 0; j < FOO; j++)      cin >> b[i][j];
 int f;
 char g;
 do
 {
  cout << "Message 2: ";    cin >> f;
  if (C(b, FOO, f))      cout << "Message 3\n";
  else      cout << "Message 5\n";
  cout << "\nMessage 4";
  cin >> g
                    ;
 }
 while (g == 'Y' || g == 'y');
}

bool C(BAR b, int n, int d){
 bool h = false;
 for (int s = 0; s < n; s++) for (int q = 0; q < n; q++)if (b[s][q] == d)h = true;
 return h;
}
```