20CS2028 – Summer 2015

Instructor: Anca Ralescu

**Programming Assignment #2**

Assigned May 28, 2015

Due on BlackBoard June 9, 2015 (11:59PM)

**This assignment is worth 30 points**

**The Element Uniqueness Problem**

A set is a collection of unique like objects. For example, the collection of SSN of students in a course is a set, while the collection of names of students in that course is not (since more than one student may have the same name). The word “like” in like objects refers in this case to the type (as in C++ type).

One way to implement sets is through arrays, ignoring the order of elements in the array: an array of size n represents a set of n elements, with no repetitions, and all arrays containing the same objects but in different order represent the same set.

Thus, with the choice of representing a set as an array we have to ensure that (1) the array contains no repeated elements (this is the element uniqueness problem), and (2) given two arrays of the same type and size, check that they represent the same set.

In this homework you are to implement the following THREE algorithms for the element uniqueness problem:

**bool** Algorithm isUnique1(A, first, last){

Input: Array A, first, last

Output:



**true** if the array contains no repeated elements

**false** if the array contains repeated elements



**if** (first >= last)

**return true**;

**if** (!isUnique1(A, first, last-1)

**return false;**

**if** (!isUnique1(A, first+1, last)

**return false;**

**return**(A[first]!=A[last])



}

**bool** Algorithm isUnique2(A, first, last){

Input: Array A, first, last

Output:

**true** if the array contains no repeated elements

**false** if the array contains repeated elements

**if** (first >= last)

**return true**;

**for** (int i = first; i < last; i++)

for (int j = i+1; j <=last; j++)

if (A[i] == A[j])

**return false;**

**return true;**

}

**bool** Algorithm isUnique3(A, first, last){

Input: Array A, first, last

Output:

**true** if the array contains no repeated elements

**false** if the array contains repeated elements

**if** (first >= last)

**return true**;

**SORT(A, first, last);**

**for** (int i = first; i < last; i++)

if (A[i] == A[i+1])

**return false;**

**return true;**

}

Furthermore, you are to do the following:

1. Implement three different sorting algorithms to be used in isUnique3. For a fixed array size evaluate which of these algorithms will produce a faster isUnique3.
2. For the same array used in 1 above implement and evaluate isUnique1, and isUnique2. Document which of these three algorithms is better (faster execution time).
3. Do the same as in point 2 above, when the input array is already sorted (and in that case comment out the sort instruction from isUnique3.
4. Now, generate arrays of increasing size n and evaluate each of your three algorithms: tabulate the results. That is, you should produce a table as shown below. Each cell contains the running time.

|  |  |  |  |
| --- | --- | --- | --- |
| n | isUnique1 | isUnique2 | isUnique3 |
| 100 |  |  |  |
| 500 |  |  |  |
| 1000 |  |  |  |
| …. |  |  |  |
| 5000 |  |  |  |

1. For each algorithm find the largest value n such that the algorithm runs in one minute or less.

For each programming assignment you should do the following:

1. **How to name the file(s) you submit**: use the last names of all students in the team followed by the assignment number. So, for example, if the team members are Brown, Smith and Johnson and the assignment number is 1 you file should be named

BrownSmithJohnson1 (with the appropriate extension, e.g., .cpp)

1. In the file containing the main function you must write at the top in a comment section the following information
   1. Author: Chuck Brown, Ben Smith, Bill Johnson
   2. Course title: Data Structures
   3. Course number: CS2028
2. Instructor: Anca Ralescu
3. TA: Suryadip Chakraborty
4. Abstract: Assignment 1 main.cpp uses the abstract interface Polygon and abstract functions area() and perimeter() to implement classes for …. and to compute their respective areas and perimeters
5. Preconditions: ….
6. Postconditions: …..
7. Credit: specify here sources that you may have used, including code.