MEDGAR EVERS COLLEGE
The City University of New York

Data Structures
CS 246 - 040
Department of Physics and Computer Science
Medgar Evers College
Exam 3

Instructions:

- **The exam requires completing tasks in a cpp file within an hour and 50 minutes.**
- **Accompanying this file is a template cpp file that you must modify. You cannot add additional libraries to or remove any libraries from the file. All other modifications are allowed.**
- **Your submissions must be submitted to the Exams directory of your github repository and/or as attachments on Google classroom under the Exam03 assessment. The file must have its accurate extension.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, AT THE BEGINNING OF YOUR SUBMISSION(S), ADD A COMMENT THAT CONSISTS OF YOUR NAME AND THE DATE

Grading:

Section	**Maximum Points**	**Points Earned**
Fundamentals	5	
Implementation	5	
Problem Solving	10	
Total	20	

Fundamentals

1. **Write ONLY what is requested in the commented section titled Fundamentals**

   a. What is the principle of a queue?

   b. What does a collision mean in hashing?

   c. What is the requirement for a map data structure?

   d. What is the load factor equal to?

   e. How does the chaining insertion method deal with collisions?

Implementation

2. **A deque data structure is a double ended queue. It allows insertions, removals and views from either end of a collection. Its implementation uses a doubly linked list and two references allows all its methods have a constant big-O runtime. These references, namely *front* and *back*, point to the first element of the list and the last element of the list respectively.**

   A *Deque* container class of a deque data structure contains following the fields

```
template<class T>
class Deque
{
  private:
  dn::Node<T>* front;
  dn::Node<T>* back;
};
```

   where `Node<T>*` **is a doubly linked pointer.**

   Given the above information, after the commented section titled Implementation, write the definition a removal method of *Deque* whose header is

```
template<class T>
void RemoveFromBack()
```

   It removes the value at the end of the doubly linked list. Make sure the deque remain consistent [references and links point to the accurate nodes].

Problem Solving

3. **After the commented section titled Problem Solving, write the definition of the int function named** `OddMedian()` **whose header is**

```
int OddMedian(Array<int>& data)
```

   Given that *data* is an unordered array of odd numbers between 200 and 300 inclusively, the function returns the median of the elements of *data* if the size of *data* is odd; otherwise, it returns 0. When a collection of numbers is ordered, the median of the numbers is the middle number when the size of the collection is odd.

4. **After the commented section titled Problem Solving, write the definition of the string function named** `Alter()` **whose header is**

```
string Alter(string str)
```

   It returns a string that is a substring *str* such that adjacent characters that are the same are removed from the string and the characters are in the reverse order. For instance, the callers `Alter("apple")` **and** `Alter("perre")` **will return** `"ela"` **and** `"p"` **respectively.**

   Hint: use a stack. You are concerned with the last valid read character and the current character read.