6. In this problem you will help National Geographic magazine collectors maintain their collection. They would like to keep track of their magazines so that the most recent issue by publication date that they have is most easily accessible to them at the top of their collection. Every issue has a unique string identifier. They would like to keep track of the total value of their collection and also be able to sell issues as necessary. In order to sell issues, they need to be able to get the cost of an issue quickly, but it is fine if searching for the actual issue is not very efficient (i.e. linear). You are to complete the implementation of the class **MagazineCollection**, with the specified runtime requirements. Hint: You may also use STL classes in your implementations if necessary. We have provided a simplified STL syntax on the first page of your exam that you may also use.

```
class MagazineCollection {
   /* Let n be the number of magazines in the collection*/
   public:
   MagazineCollection();
   /* constructor initializes an empty collection with zero value. Worst case runtime: O(1)*/

   ~MagazineCollection (); // destructor

   void addMagazine(string issue, int value);
   /* adds the magazine issue with given value. Assume that this function will be called by the
      collectors with magazine issues in the order that the issues are released by publication
      date. Assume value is greater than 0. If the magazine is already in the collection, this
      function should do nothing. Worst case runtime: O(n)  */

   int getMagazineValue(string issue) const;
   /* returns the value of any magazine in the collection. Returns 0 if the issue is not in the
       collection. Worst case runtime: O(log n) */

   int getValueOfRecentM(int m) const;
   /* returns value of the m most recent magazine issues in the collection. If n < m, return
       value of entire collection. Worst case runtime: O(n log n) */

   void sellMagazine(string issue);
   /* this function should remove the magazine with the given issue identifier from the
      collection of magazines and its value from the collection value. If the magazine issue is
      not in the collection, this function should do nothing. Worst case runtime: O(n)*/

   int getCollectionValue() const;
    /* Return the value of the entire magazine collection. Worst case runtime: O(1) */

   private:
   // feel free to add private member variables or functions
```

std::map<string, int> values;
stack<string> issues;
int total;

```
};
```

```
MagazineCollection::MagazineCollection()   /* Worst case runtime: O(1) */
{
      total = 0;

}
MagazineCollection::~MagazineCollection () {
        // Nothing needed
}
void MagazineCollection::addMagazine(string issue, int value){ /*Worst case runtime: O(n)*/
     if (getMagazineValue(issue) != 0) return;
      issues.push(issue);
      values.insert(issue, value);
      total += value;


}
int MagazineCollection::getMagazineValue(string issue) const  /*Worst case runtime: O(log n)*/
{
        if (values.find(issue) == values.end()) {
          return 0;
      } else {
            return values[issue];
      }
}
int MagazineCollectioon::getCollectionValue() const {  /*Worst case runtime: O(1)*/
      return total;

}
```

```
int MagazineCollection::getValueOfRecentM(int m) const /*Worst case runtime: O(n*log(n))*/
{
    int mtot = 0;
    stack<string> t;
    while( (! issues.empty()) && m>0){
        string cur= issues.top();
        t.push(cur);
        issues.pop();
        mtot += values[cur]; m--;
    }
    while ( ! t.empty()){
        string cur = t.top();
        issues.push(cur);
        t.pop();
    }
    return mtot;
}
void MagazineCollection::sellMagazine(string issue) /*Worst case runtime: O(n)*/
{
    if (getMagazineValue(issue)==0) return;
    stack<string> t;
    string cur = issues.top();
    while (cur != issue){
        t.push(cur);
        issues.pop();
        cur = issues.top();
    }
    issues.pop();
    total -= values[issue];
    values.erase[issue];
    while (! t.empty()){
        issues.push(t.top());
        t.pop();
    }
}
```