1. *(1 pt)* Write a function `bar` that takes a positive integer named `sales` as an argument and outputs a line in a bar graph. Each asterisk in the output should represent $100 of sales. The function should output 5 asterisks (followed by a newline character) for `sales = 590`.

2. *(3 pts)* Given a linked list class defined as follows:

   ```
   class NumberList
   {
       private:
           struct Node
           {
               double value;
               struct Node *next;
           };
           ListNode *head; //points to the first node
       public:
           NumberList();
   }
   ```

   Write a member function `mergeItems` that takes an array of `doubles` and its size as arguments. Assume the linked list is in sorted order. The function should insert each element of the argument array into the list in its appropriate place so that the list remains sorted. NOTE: you may assume that the linked list has at least 2 nodes and all of the numbers in the array are greater than the first element and less than the last element in the linked list.
3. (2 pts) Addition of positive integers can be defined using the following recursive equation:

\[ x + 0 = x; \]
\[ \text{for } y > 0, \ x + y = (+x) + (−−y) \]

Write a recursive function `ladd` that takes two integer arguments (say `x`, and `y`) and returns the sum `z = x + y`. The result `z` must be computed using the recursive equation without using the `+` operator (you may use `++`).

Note: The actual recursion, as defined by the Peano set of axioms, uses the successor and predecessor operations. In this problem, however, we represent the successor and the predecessor of an integer `i` by `(++i)` and `(--i)` respectively.

4. (4 pts) Given the following `Set` class that represents a collection of unique integers:

```cpp
class Set {
private:
    int * elements; //pointer to dynamically allocated array
    int num; //the number of elements in the set
public:
    Set () { elements = NULL; num = 0; }
    void add(int); // adds an int to the set if not already there
    bool find(int); // returns true if the argument is in the set
    Set operator+ (Set); // returns the union of the 2 sets.
    ~Set(); // destroys the set
};
```

Note that `num` also stores the actual size of the array.

Implement the four undefined methods from the `Set` class. Note:
- You must expand the array by one element when you add an element.
- The elements in the set must be unique.