Problem 1: Arrays

Function `remove_equal()` takes as arguments an array of integers, `list` and an integer `x` and removes all occurrences of `x` from `list`. When an element is removed, the remaining elements are shifted over and any opened slots at the end of the array are filled with zeros. For example, if `list` is `{1,4,3,1,6,4,1,1}` and `x` is 1, then `remove_equal(list, x)` will cause the list to become `{4,3,6,4,0,0,0,0}`. Write the implementation of `remove_equal()`. You may assume that the array has `SIZE` elements.

```c
void remove_equal(int list[], int x) {
    for (int i=0; i<SIZE; i++) {
        if (list[i] == x) {
            for (int j=i; j<SIZE-1; j++) // shift rightmost elements to the left
                list[j] = list[j+1];
            list[SIZE-1] = 0;
            i--; // re-examine the current location since an x may have shifted over
        }
    }
}
```

Notes:
- The inner loop should allow `j` to only go up to `SIZE-2`, so that `list[j+1]` goes only up to `list[SIZE-1]` which is the last element of the array.
- In general, it is dangerous to modify the loop control variable (`i`) in the body of the loop. Avoid doing that unless you are absolutely certain it will not cause an endless loop.

Problem 2: Pointers

Write a `main()` function that can be used to test the `remove_equal()` that you wrote in problem 1. Your `main()` should create a dynamically allocated array, use `cin` to fill it with integers and then test `remove_equal()`. Don't forget to print the contents of the resulting array. You must use a dynamically allocated array and only pointer syntax.

```c
int main () {
    int *list = new int[SIZE];
    for (int i=0; i<SIZE; i++)
        cin >> *(list+i);
    remove_equal(list, 1);
    for (int i=0; i<SIZE; i++)
        cout << *(list+i) << " ";
    delete [] list;
    return 0;
}
```
An alternative answer that uses a temporary pointer to hold the beginning of the list and then traverses the elements by incrementing the `list`:

```cpp
int main () {
    int *list = new int[SIZE];
    int *ptr_holder = list;

    for (int i=0; i<SIZE; i++) {
        cin >> *list;
        list++;
    }
    list = ptr_holder;

    remove_equal(list, 1);

    for (int i=0; i<SIZE; i++) {
        cin >> *list;
        list++;
    }
}
```