And here is a simple annotated suffix sorting code:

```
#include <iostream>
#include <string>
#include <algorithm>
#include <ctime>
using namespace std;
#define N 10000000

char str[N];
int H = 0, Bucket[N], nBucket[N], c;

struct Suffix{
        int idx; // Suffix starts at idx, i.e. it's str[ idx .. L-1 ]
        bool operator<(const Suffix& sfx) const
        // Compares two suffixes based on their first 2H symbols,
        // assuming we know the result for H symbols.
        {
                if(H == 0) return str[idx] < str[sfx.idx];
                else if(Bucket[idx] == Bucket[sfx.idx])
                        return (Bucket[idx+H] < Bucket[sfx.idx+H]);
                else
                        return (Bucket[idx] < Bucket[sfx.idx]);
        }
        bool operator==(const Suffix& sfx) const
        {
                return !(*this < sfx) && !(sfx < *this);
        }
} Pos[N];

int UpdateBuckets(int L)
{
        int start = 0, id = 0, c = 0;
        for(int i = 0; i < L; i++)
        {
                /*
                        If Pos[i] is not equal to Pos[i-1], a new bucket has started.
                */
                if(i != 0 && !(Pos[i] == Pos[i-1]))
                {
                        start = i;
                        id++;
                }
                if(i != start) // if there is bucket with size larger than 1, we should continue ...
                        c = 1;
                nBucket[Pos[i].idx] = id; // Bucket for suffix starting at Pos[i].idx is id ...
        }
        memcpy(Bucket, nBucket, 4 * L);
        return c;
}

void SuffixSort(int L)
{
        for(int i = 0; i < L; i++) Pos[i].idx = i;
        // H == 0, Sort based on first Character.
        sort(Pos, Pos + L);
        // Create initial buckets
        c = UpdateBuckets(L);
        for(H=1;c;H *= 2) {
                // Sort based on first 2*H symbols, assuming that we have sorted based on first H character
                sort(Pos, Pos+L);
                // Update Buckets based on first 2*H symbols
                c = UpdateBuckets(L);
        }
}
```

```
int main()
{
        cin >> str;
        int L = strlen(str) + 1;
        int cl = clock();
        SuffixSort(L);
        cerr << (clock() - cl) * 0.001 << endl;
        for(int i = 0; i < L; i++)
                cout << "'" << str + Pos[i].idx << "'" << endl;
        return 0;
}
```