```
// Die Funktion sortiert einen Vektor, den sie als Referenz erhält
void RadixSort(std::vector<unsigned int> &_vector) {
    // Variable, welcher Radix gesucht wird
    unsigned int digitplace = 1;
    // Ein neuer Vector zur Zwischenspeicherung wird erstellt
    std::vector<unsigned int> result(_vector.size());
    // Maximum des Vektors
    unsigned int max = GetMax(_vector);
    // Prüfen, ob alle signifikanten Stellen der größten Zahl durchlaufen wurden
    while(max/digitplace > 0){
        // Speicherort für die Zähler
        std::vector<int> counters_array(10, 0);
        // Zählen, wie oft jede Ziffer vorkommt
        CountingRoutine(_vector, digitplace, 0, _vector.size(), counters_array,
0);
        // Errechnen, wohin Zahlen mit der jeweiligen Ziffer plaziert werden
müssen
        PrefixSum(counters_array);
        // Aufbauen des neuen, teilweise sortierten Vectors
        for (int i = int(_vector.size()) - 1; i >= 0; i--) {
            // Bestimmten des Radix der aktuellen Zahl im alten Vektor
            unsigned int digit = (_vector[i]/digitplace) % 10;
            // Verringen der Position, wohin diese Zahl sohl
            counters_array[ digit ]--;
            // Einsetzen der Zahl im neuen Vektor
            result[ counters_array[ digit ] ] = _vector[i];
        }
        // Kopieren der Elemente in den ursprünglichen Vektor
        // zur Vorbereitung des nächsten Durchlaufs
        for (unsigned long i = 0; i < _vector.size(); ++i) {
            _vector[i] = result[i];
        }
        // Prüfen, ob es einen Integer overflow gibt,
        // also ob die letzte Ziffer eines 32 bit Integers
        // geprüft wurde
        unsigned long multiple = long(digitplace) * 10;
        if(multiple > std::numeric_limits<unsigned int>::max()){
            break;
        } else {
            digitplace = multiple;
        }
    }
}
```