Q1 b) Implement Quick Sort Algorithm. Analyze its time complexity.

TIME COMPLEXITY:

1. Best Case: O(n * log(n))
2. Worst Case: O(n ^ 2)
3. Average Case: O(n * log(n))

SPACE COMPLEXITY: O(1)

```
#include <iostream>
#include <vector>
#include <random>


using namespace std;


int partition(vector<int> & arr, int start, int end)
{
    int pivot = arr[end];
    int smallIndex = start;
    for(int i = start; i < end; i++)
        if(arr[i] <= pivot)
            swap(arr[smallIndex++], arr[i]);
    swap(arr[smallIndex], arr[end]);
    return smallIndex;
}

void quickSort(vector<int> & input, int start, int end)
{
    if(start >= end)
        return;
    int p = partition(input, start, end);
    quickSort(input, start, p - 1);
    quickSort(input, p + 1, end);
}

int main()
{
    vector<int> A(100);
    for(int i = 0; i < 100; ++i)
        A[i] = rand() % 500;
    cout << "INITIAL STATE OF ARRAY:\n";
    for(int i = 0; i < 100; ++i)
        cout << A[i] << " ";
    cout << endl << endl;
    quickSort(A, 0, int(A.size()) - 1);
    cout << "SORTED ARRAY:\n";
    for(int i = 0; i < 100; ++i)
        cout << A[i] << " ";
    cout << endl;
}
```

OUTPUT:

```
INITIAL STATE OF ARRAY:
307 249 73 158 430 272 44 378 423 209 440 165 492 42 487 3 327 229 340 112
 303 169 209 157 60 433 99 278 316 335 97 326 12 267 310 133 479 149 79 321
 467 172 393 336 485 245 228 91 194 357 1 153 208 444 168 490 124 196 30
 403 222 166 49 24 301 353 477 408 228 433 298 481 135 13 365 314 63 36 425
 169 115 94 129 1 17 195 105 404 451 298 188 123 5 382 252 66 216 337 438
 144

SORTED ARRAY:
1 1 3 5 12 13 17 24 30 36 42 44 49 60 63 66 73 79 91 94 97 99 105 112 115
 123 124 129 133 135 144 149 153 157 158 165 166 168 169 169 172 188 194
 195 196 208 209 209 216 222 228 228 229 245 249 252 267 272 278 298 298
 301 303 307 310 314 316 321 326 327 335 336 337 340 353 357 365 378 382
 393 403 404 408 423 425 430 433 433 438 440 444 451 467 477 479 481 485
 487 490 492
Program ended with exit code: 0
```