Q5. Given a large single dimensional array of integers, write functions for sliding window filter with maximum, minimum, median, and average to generate an output array. The window size should be an odd integer like 3, 5 or 7. Explain what you will do with the boundary values. (Use zero-padding in both ends of the arrays as necessary.)

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

void zeroPadArray(int\* input, int n, int k, int\* padded) {

int pad = k / 2;

for (int i = 0; i < pad; i++)

padded[i] = 0;

for (int i = 0; i < n; i++)

padded[i + pad] = input[i];

for (int i = 0; i < pad; i++)

padded[n + pad + i] = 0;

}

void slidingWindowMax(int\* input, int n, int k, int\* output) {

int padArray[n + k - 1];

zeroPadArray(input, n, k, padArray);

for (int i = 0; i < n; i++) {

int maxVal = INT\_MIN;

for (int j = 0; j < k; j++)

if (padArray[i + j] > maxVal)

maxVal = padArray[i + j];

output[i] = maxVal;

}

}

void slidingWindowMin(int\* input, int n, int k, int\* output) {

int padArray[n + k - 1];

zeroPadArray(input, n, k, padArray);

for (int i = 0; i < n; i++) {

int minVal = INT\_MAX;

for (int j = 0; j < k; j++)

if (padArray[i + j] < minVal)

minVal = padArray[i + j];

output[i] = minVal;

}

}

void slidingWindowMedian(int\* input, int n, int k, int\* output) {

int padArray[n + k - 1];

zeroPadArray(input, n, k, padArray);

int window[k];

for (int i = 0; i < n; i++) {

for (int j = 0; j < k; j++)

window[j] = padArray[i + j];

qsort(window, k, sizeof(int), compare);

output[i] = window[k / 2];

}

}

void slidingWindowAverage(int\* input, int n, int k, int\* output) {

int padArray[n + k - 1];

zeroPadArray(input, n, k, padArray);

for (int i = 0; i < n; i++) {

int sum = 0;

for (int j = 0; j < k; j++)

sum += padArray[i + j];

output[i] = sum / k;

}

}

int compare(const void\* a, const void\* b) {

return (\*(int\*)a - \*(int\*)b);

}