int k = 1;

*// #pragma unroll*

*// for (int k = 1; k <= 12; k++) {*

*// force the 3k+1 th iteration to be repeated*

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

if (((k % 3) == 1) && (k != 1)) {

*// #pragma unroll*

*// for (int j = 1; j <= 2; j++) {*

*// beta<0 anti-clockwise rotation*

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

*// }*

} else {

if (beta < 0) {

x\_new = x - (y >> k);

y -= x >> k;

beta += index\_trigo;

}

*// beta>0 clockwise rotation*

else {

x\_new = x + (y >> k);

y += (x >> k);

beta -= index\_trigo;

}

x = x\_new;

}

k++;

*// }*