The problem I faced is to use an elegant way to write the last function, because I don’t want to use new array. I used a method that is similar to quick search. I didn’t use new arrays in all the functions.

//testing reduplicate

string a[7] = {"nihao", "wohenhao", "zhehenbucuo", "zhenbucuo", "woaics", "nimenhaoa", "taibangle"};

//when n is negative

assert(reduplicate(a, -1) == -1);

// when n is zero

assert(reduplicate(a,0) == 0);

assert(a[0] == "nihao" && a[1] == "wohenhao" && a[2] == "zhehenbucuo" && a[3] == "zhenbucuo"

&& a[4] == "woaics" && a[5] == "nimenhaoa" && a[6] == "taibangle");

// when n is non zero, for all elements in the array

string b[7] = {"nihao", "wohenhao", "zhehenbucuo", "zhenbucuo", "woaics", "nimenhaoa", "taibangle"};

assert(reduplicate(b, 7) == 7);

assert(b[0] == "nihaonihao" && b[1] == "wohenhaowohenhao" && b[2] == "zhehenbucuozhehenbucuo" && b[3] == "zhenbucuozhenbucuo"

&& b[4] == "woaicswoaics" && b[5] == "nimenhaoanimenhaoa" && b[6] == "taibangletaibangle");

// when n is non zero, for part elements in the array

string c[7] = {"nihao", "wohenhao", "zhehenbucuo", "zhenbucuo", "woaics", "nimenhaoa", "taibangle"};

assert(reduplicate(c, 5) == 5);

assert(c[0] == "nihaonihao" && c[1] == "wohenhaowohenhao" && c[2] == "zhehenbucuozhehenbucuo" && c[3] == "zhenbucuozhenbucuo"

&& c[4] == "woaicswoaics");

// testing locate with the element in the array

string d[4] = {"nihao", "wohenhao", "zhehenbucuo", "zhenbucuo"};

assert(locate(d, 4, "nihao") == 0 && locate(d, 4, "wohenhao") == 1 && locate(d, 4, "zhehenbucuo") == 2 && locate(d, 4, "zhenbucuo") == 3);

//testing locate with the element that's not in the array

assert(locate(d, 4, "mlgb") == -1);

//testing locate with partial range

assert(locate(d, 3, "nihao") == 0 && locate(d, 3, "zhehenbucuo") == 2 && locate(d, 3, "zhenbucuo") == -1);

//testing locate with negative n

assert(locate(d, -1, "jello") == -1);

//testing locate when n is 0

assert(locate(d, 0, "nihao") == -1);

//testing locationOfMax with valid input

string e[4] = {"Nihao", "nihao", "nIhao", "niHao"};

assert(locationOfMax(e, 4) == 1);

//testing locationOfMax with non positive input

assert(locationOfMax(e, -1) == -1 && locationOfMax(e, 0) == -1);

//testing cirlceLeft

string f[4] = {"a", "b", "c", "d"};

assert(circleLeft(f, 4, 0) == 0 && f[0] == "b" && f[1] == "c" && f[2] == "d" && f[3] == "a");

string g[4] = {"a", "b", "c", "d"};

assert(circleLeft(g, 4, 1) == 1 && g[0] == "a" && g[1] == "c" && g[2] == "d" && g[3] == "b");

string h[4] = {"a", "b", "c", "d"};

assert(circleLeft(h, 4, 3) == 3 && h[0] == "a" && h[1] == "b" && h[2] == "c" && h[3] == "d");

assert(circleLeft(h, 3, 0) == 0 && h[0] == "b" && h[1] == "c" && h[2] == "a" && h[3] == "d");

assert(circleLeft(h, 0, 0) == -1 && circleLeft(h, -1, 0) == -1);

//testing enumerate runs

string i[6] = {"a", "a", "a", "n", "n", "n"};

assert(enumerateRuns(i, 6) == 2);

string j[2] = {"", ""};

assert(enumerateRuns(j, 2) == 1);

// testig flip

string k[4] = {"a", "b", "c", "d"};

assert(flip(k, 4) == 4 && k[0] == "d" && k[1] == "c" && k[2] == "b" && k[3] == "a");

// testing flip when n is non-positive

assert(flip(k, -1) == -1 && flip(k, 0) == 0 && k[0] == "d" && k[1] == "c" && k[2] == "b" && k[3] == "a");

// tetsing locate difference

string l1[4] = {"a", "b", "c", "d"};

string l2[4] = {"a", "a", "a", "a"};

string l3[4] = {"", "", "", "",};

assert(locateDifference(l1, 4, l2, 4) == 1);

// when n is zero or negative

assert(locateDifference(l1, 0, l2, 0) == 0 && locateDifference(l1, -2, l2, -2) == -1);

assert(locateDifference(l1, 4, l3, 4) == 0);

//testing subsequence

string m1[5] = {"a", "b", "c", "d", "e"};

string m2[3] = {"a", "b", "c"};

string m3[3] = {"a", "c", "b"};

string m4[4] = {"f", "h", "i", "j"};

assert(subsequence(m1, 5, m2, 3) == 0 && subsequence(m1, 5, m3, 3) == -1 && subsequence(m1, 5, m4, 4) == -1);

assert(subsequence(m1, 5, m2, 0) == 0 && subsequence(m1, 0, m2, 0) == 0 && subsequence(m1, -1, m2, -1) == -1);

//testing locateAny

string n1[5] = {"a", "b", "c", "d", "e"};

string n2[2] = {"a", "vf"};

string n3[2] = {"3", "e"};

assert(locateAny(n1, 5, n2, 2) == 0 && locateAny(n1, 5, n3, 2) == 4);

assert(locateAny(n1, 0, n2, 0) == -1 && locateAny(n1, 5, n3, 0) == -1 && locateAny(n1, 0, n3, 2) == -1 && locateAny(n1, -1, n3, -1) == -1);

//testing separate

string o1[5] = {"a", "b", "z", "d", "e"};

string o2[5] = {"a", "b", "y", "d", "e"};

string o3[5] = {"a", "b", "z", "d", "e"};

assert(separate(o1, 5, "d") == 2);

assert(separate(o2, 5, "z") == 5);

assert(separate(o3, 5, "a") == 0);