**Report**

**Summary:**

The most notable obstacle I met in the homework is the last function. I managed to realize it by sorting the elements in an ascending order in the first place and then finding the return value. The idea of sorting things makes the process much easier.

**Test data:**

**Function 1: int appendToAll(string a[], int n, string value)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=0

value=”2333”

Check if n is equal to zero.

(2):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=-2

value=”2333”

Check if n is negative.

(3):

string s[]={””,””,””,””,””,””}

int n=0

value=”test”

Check if array s consists of empty strings.

(4):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

value=”2333”

Check if the function works in normal condition.

**Function 2: int lookup(const string a[], int n, string target)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n =0;

target=”grace”

Check if n is zero.

(2):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n =6;

target=”boy\_next\_door”

Check if target is not in the array.

(3):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n =1;

target=”grace”

Check if target is beyond the range given by n.

(4):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n =5;

target=”grace”

Check if target is within the range given by n.

**Function 3: int positionOfMax(const string a[], int n)**

(1):

string s[]={””,””,””,””,””,””}

int n=0

Check if n is zero.

(2):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”zebra”,”enigma”,”cherry”}

int n=6

Check if there are repeated strings in the array.

(3):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

Check if the function works in normal condition.

**Function 4: int rotateLeft(string a[], int n, int pos)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

int pos = 0

Check if the selected string is the first one in the array.

(2)

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

int pos = 5

Check if the selected string is the last one in the array.

(3) string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=5

int pos =2

Check if the function works in normal condition.

**Function 5: int countRuns(const string a[], int n)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

Check if these is no repeated element.

(2):

string s[]={”michael”,” michael”,”grace”,” grace”,” grace”,”cherry”}

int n=6

Check if these is repeated elements.

**Function 6: int flip(string a[], int n)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=6

Check if the array contains an even number of elements.

(2):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n=7

Check if the array contains an odd number of elements.

(3):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n=1

Check if only one element is taken into account.

**Function 7: int differ(const string a1[], int n1, const string a2[], int n2)**

(1):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=7

check if the two arrays are identical.

(2):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”gebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=7

check if the two arrays are different.

(3):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,}

int n1=7

int n2=3

check if the two arrays are identical but values of n differ.

(4):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”tree”}

int n1=7

int n2=3

check if both the arrays and values of n differ.

**int subsequence(const string a1[], int n1, const string a2[], int n2)**

(1):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=7

check if the two arrays are identical.

(2)

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=0

int n2=7

check if n1 is zero.

(3)

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=0

check if n2 is zero.

(4)

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=0

int n2=0

check if both n1 and n2 are zero.

(5)

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”rachel”,”cherry”,”ben”}

int n1=7

int n2=3

Check if s1 doesn’t contain s2.

(6):

string s1[]={“tree”,”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=4

Check if s1 contains s2.

**Function 9: int lookupAny(const string a1[], int n1, const string a2[], int n2)**

(1):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={“”}

int n1=7

int n2=1

check if nothing in s1 matches with anything in s2.

(2):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={“gg”,”mm”,”michael”}

int n1=7

int n2=3

check if something in s2 matches with s1.

(3):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={“”}

int n1=0

int n2=1

check if n1 is zero

(3):

string s1[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

string s2[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”,”miguel”}

int n1=7

int n2=0

check if n2 is zero

**function 10: int divide(string a[], int n, string divider)**

(1):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=0

divider = “Assassin’s\_creed”

check if the divider is smaller than all the elements in s.

(2):

string s[]={”michael”,”andrews”,”grace”,”yebra”,”enigma”,”cherry”}

int n=0

divider = “zelda”

check if the divider is larger than all the elements in s.

(3):

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=0

divider = “darksouls”

check if the divider is in the middle of the elements in s.

(4)

string s[]={”michael”,”andrews”,”grace”,”zebra”,”enigma”,”cherry”}

int n=0

divider = “grace”

check if the divider is one element in s.