#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\*\* split\_string(char\*);

// Complete the designerPdfViewer function below.

int designerPdfViewer(int h\_count, int\* h, char\* word) {

int area;

int len;

int i;

len = strlen(word);

int conv[len];

for(i=0;i<len;i++)

{

int temp = word[i] - 'a';

conv[i] = h[temp];

}

//finding max

int max=0;

for(i=0;i<len;i++)

{

if(conv[i] > max)

{

max = conv[i];

}

}

area = max\*len;

return area;

}

int main()

{

FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

char\*\* h\_temp = split\_string(readline());

int\* h = malloc(26 \* sizeof(int));

for (int i = 0; i < 26; i++) {

char\* h\_item\_endptr;

char\* h\_item\_str = \*(h\_temp + i);

int h\_item = strtol(h\_item\_str, &h\_item\_endptr, 10);

if (h\_item\_endptr == h\_item\_str || \*h\_item\_endptr != '\0') { exit(EXIT\_FAILURE); }

\*(h + i) = h\_item;

}

int h\_count = 26;

char\* word = readline();

int result = designerPdfViewer(h\_count, h, word);

fprintf(fptr, "%d\n", result);

fclose(fptr);

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) { break; }

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

size\_t new\_length = alloc\_length << 1;

data = realloc(data, new\_length);

if (!data) { break; }

alloc\_length = new\_length;

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

}

data = realloc(data, data\_length);

return data;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces - 1] = token;

token = strtok(NULL, " ");

}

return splits;

}