ENCRYPTION (STRING)

An english text needs to be encrypted using the following encryption scheme. First the spaces are removed from the text.

Let L be the length of the text.

Then, characters are written into a grid, whose rows and columns have the following constraints.

$\lfloor \sqrt{L} \rfloor \leq row \leq column \leq \lceil \sqrt{L} \rceil$, where $\lfloor x \rfloor$ is the floor function and $\lceil x \rceil$ is the ceiling function.

Example

S = if man was meant to stay on the ground god would have given us roots.

After removing spaces, the string is 54 characters long.

$\sqrt{54}$ is b/w 7 and 8, so it is written in a form of grid with 7 rows and 8 columns.

8 columns →

7 rows ↓

```
if manwas
meanttos
tayonthe
groundgo
dwouldha
vegivenu
sroots
```

- Ensure that row x columns $\geq L$.
- If multiple grids satisfy the above conditions, choose the one with the minimum area i.e rows x columns.

The encoded message is obtained by displaying the characters of each column, with a space b/w column texts. The encoded message for the grid above is :

imtgdvs fearwer mayoogo anouuio ntnnlvt

wttddes aohghn sseoau.

Example

→ sample input

haveaniceday → L = 12, $\sqrt{12}$ is b/w 3 and 4.

Rewritten with 3 rows and 4 columns.

h a v e
a n i c
e d a y

sample output ← hae and via ecy

output

C code

```
void encryption (char input[]) {
    int len = strlen (input);
    char text[81];
    int index = 0;

// Remove spaces from input
    for (int i=0; i<len; i++) {
        if(input[i] != ' ') {
            text[index++] = input[i];
        }
    }
    text[input] = '\0'; // NULL terminate the string
    len = index;

// calculate rows and columns
    int rows = (int) floor (sqrt(len));
    int cols = (int) ceil (sqrt(len));
    if (row * cols < len) {
        rows++;
    }

// print encrypted message
    for (int col = 0; col < cols; col++) {
        for (int row = 0; row < rows; row++) {
            int idx = row * cols + col;
            if (idx < len) {
                putchar (text[idx]);
            }
        }
    }
```

```
        if (col != cols - 1) {
            putchar(' ');
        }
    }
    putchar('\n');
}

int main() {
    char input[81];
    // Read input string
    fgets(input, sizeof(input), stdin);

    // Remove the newline character
    for (int i = 0; input[i] != '\0'; i++) {
        if (input[i] == '\n') {
            input[i] = '\0';
            break;
        }
    }

    // call the encryption function
    encryption(input);

    return 0;
}
```