Dirilididi - Passo a passo

Primeiro acesso:

1. Vá até http://dirlididi.com
2. Faça login com sua conta @ccc
3. Cadastre-se na sua turma usando o código correspondente (turma 1: QLzYfjiZk, turma 2: UGi4PgMzY)
4. Copie o seu token de identificação. Ele será usado para a submissão dos laboratórios

Durante os laboratórios:

1. Crie uma pasta local para armazenar seus arquivos
2. Faça o download do arquivo http://dirlididi.com/tools/dirlididi.py para sua pasta local
3. Acesse o link da questão do laboratório
4. Crie o arquivo de resposta do laboratório com seu código fonte
5. Compile o arquivo e submeta o exercício usando a seguinte linha de comando (usando sua pasta local como ponto de partida):
   python dirlididi.py submit <problem_key> <token> <filename>
   <problem_key> : identificador da questão
   <token> : identificador do usuário
   <filename> : nome do arquivo compilado

Para laboratórios de C/C++:

1. compile seu código fonte .cpp : **gcc <filename>.cpp -o <filename>**
2. Submeta sua respota: **python dirlididi.py submit <problem_key> <token> <filename>.exe**

Para laboratórios de Haskell:

3. compile seu código fonte .cpp : **ghc <filename>.hs <filename>**
4. Submeta sua respota: **python dirlididi.py submit <problem_key> <token> <filename>.exe**

Para laboratórios de Prolog:

5. Submeta sua respota: **python dirlididi.py submit <problem_key> <token> <filename>.pl**

Direcionamento geral dos laboratórios:
Para que seu programa passe nos testes, este deve receber entradas como strings, tratá-las adequadamente e reportar as saídas usando a saída padrão do sistema. Veja o exemplo abaixo, para diferentes linguagens, de um programa que recebe um valor inteiro como entrada e retorna seu dobro:
Questão: http://dirlididi.com/client/index.html#ide/NnUrN9Q8W

- C/C++

```
#include <stdio.h>
int main(){
        int a;
        scanf("%d", &a);
        printf("%d", a*2);

}
```

- Haskell

```
doubleMe :: Int -> Int
doubleMe x = x * 2

main = do
   input <- getLine
   let num = doubleMe (read input)
   print num
```

- Prolog

```
:- initialization main.

main:-
   repeat,
   read_line_to_codes(user_input, X3),
   string_to_atom(X3,X2),
   atom_number(X2,X),
   Y is X * 2,
   write(Y),nl,
   halt(0).
```