Input Format

Alemia este o aplicație prin care se realizează predicția notelor studenților la o materie care conține surse de cod de programare.

Ca date de intrare: o arhivă .zip în care există sunt una sau mai multe surse de cod (.cpp, .h, .java, .html, .py, .ipynb, .css etc.).

```
for filename in glob.iglob(src_dir + "*/.*", recursive=True):
        fields = filename.split("/")

        my_dir = ""
        for i in range(len(fields)):
            if fields[i] == dest_fields[3]:
                my_dir = fields[i + 1]

        if all(x not in my_dir for x in newfiles):
            newfiles.append(my_dir)

        create_dir(dest_dir + "/" + my_dir)
        create_dir(dest_dir + "/" + my_dir + "/text")
        create_dir(dest_dir + "/" + my_dir + "/headers")
        create_dir(dest_dir + "/" + my_dir + "/sources")
        create_dir(dest_dir + "/" + my_dir + "/rest")

        if (filename.endswith(".txt")):
            shutil.copy2(filename, dest_dir + my_dir + "/text/" + fields[-1])
        elif (filename.endswith(".h")):
            shutil.copy2(filename,
                         dest_dir + my_dir + "/headers/" + fields[-1])
        elif (filename.endswith(".cpp")):
            shutil.copy2(filename,
                         dest_dir + my_dir + "/sources/" + fields[-1])
        elif ("Debug" in filename or "DEBUG" in filename or "/." in filename) == False:
            shutil.copy2(filename, dest_dir + my_dir + "/rest/" + fields[-1])
```

Prin secvența de cod de mai sus se realizează clasificarea fișierelor sub diferite forme.

```
inheritance_pattern = re.compile(r"([A-Z])\w+ : ([a-z]\w+)")
virtual_pattern = re.compile(r"virtual ([a-z]\w+)")
static_pattern = re.compile(r"\W*(static)\W*")
global_pattern = re.compile(r"\W*(global)\W*")
public_pattern = re.compile(r"\W*(public)\W*")
private_pattern = re.compile(r"\W*(private)\W*")
protected_pattern = re.compile(r"\W*(protected)\W*")
define_pattern = re.compile(r"^#define*")
template_pattern = re.compile(r"\W*(template)\W*")
stl_pattern = re.compile(r"\W*(std)\W*")
namespace_pattern = re.compile(r"\W*(namespace)\W*")
comments_pattern = re.compile(r"\W*(/\*)|(//)\W*")
enum_pattern = re.compile(r"\W*(enum)\W*")
struct_pattern = re.compile(r"W*(stuct)\W*")
function_pattern = re.compile(r"\W*(\(\))\W*")
```

După clasificarea fișierelor în foldere, se executa o căutare de tip regex după diferite elemente de interes precum numărul de clase, numărul de funcții, numărul de templeturi, structuri etc. Predicția unei note se va realiza în funcție de frecvența apariției caracteristicilor enumerate anterior.

Datorită mecanismelor de clasificare a tipurilor de fișiere după extensie, chiar dacă utilizatorul va introduce o arhivă ce conține fișiere neconforme, execuția programului nu va fi afectată.