# Závazné pokyny pro funkcionální projekty FLP 2019/2020

Libor Škarvada
iskarvada@fit.vutbr.cz

## 1 Požadavky na zdrojový text

Funkcionální část bude vypracována v jazyce Haskell. Při opravování použijte kompilátor GHC se standardními knihovnami (balík `base`). Své řešení si můžete otestovat na serveru `merlin`.

- Úvodní řádky zdrojových souborů musí obsahovat název projektu, login a jméno autora, rok řešení.
- Definice každé hodnoty na globální úrovni a každého typu a typové třídy začíná stručným a výstižným komentářem popisujícím danou hodnotu, typ nebo třídu.
- Součástí definice každé hodnoty je typová anotace.
- Volte vhodné datové typy, a to i s ohledem na efektivnost algoritmu.
- Vyhýbejte se parciálním funkcím, totální funkce jsou bezpečnější.
- Nepoužívejte hluboce zanořované konstrukce (if-výrazy, case, where apod).
- Nepoužívejte nestandardní funkce a akce, které obcházejí bezpečné otypování nebo obcházejí monadické zapouzdření vedlejších efektů – vesměs jsou to hodnoty obsahující ve svém jméně slovo „unsafe“.
- Neopakujte stejné části kódu, dodržujte princip DRY.
- Použijte nástroj `hlint`, který vás upozorní na nedostatky v kódu a na jeho části, které by šly napsat lépe.
- Pište přehledně, odsazujte jednotným stylem a mezerami (nikoli tabulátory), nedělejte řádky delší než 100 znaků.

Konkrétní požadavky jsou specifikovány v příslušné variantě zadání.

## 2 Soubor s řešeným projektem

Svá řešení odevzdávejte v předepsaném tvaru (zkomprimovaný archiv zip, viz Závazné pokyny pro projekty) prostřednictvím WIS do datového skladu předmětu FLP.

Odevzdaný projekt musí obsahovat zdrojové texty v Haskellu. V hlavní složce musí být soubor `Makefile`, který program přeloží a výsledný binární kód umístí také do hlavní složky. Dále se doporučuje přiložit stručný popis všeho, co nebylo dořešeno nebo naopak co bylo implementováno nad rámec zadání. Například soubor `flp-fun-xlogin00.zip` po rozbalení obsahuje:

```
Makefile
src
├── Main.hs
├── Minimize.hs
├── ParseInput.hs
└── Types.hs
doc
├── README.md
└── test-description.txt
test
├── test01.in
├── test01.out
├── test02.in
└── test02.out
```

## 3 Implementace

Součástí řešení bude soubor `Makefile` (projekty budou překládány pomocí `gmake`), ve kterém lze nastavit případné parametry překladu. Pokud soubor pro sestavení cílového programu nebude obsažen nebo se na jeho základě nepodaří sestavit cílový program, nebude projekt hodnocen. Cílový program pojmenujte podle pokynů v příslušné variantě zadání.

Pokud není řečeno jinak, vytvořte program ve formě konzolové aplikace. Nepožaduje se grafické uživatelské rozhraní.

## 4 Samostatné zadání

Pokud máte zájem pracovat na něčem zajímavějším než jsou generická společná zadání, použijte variantu `custom`, formulujte zadání a kontaktujte cvičícího.

## 5 Konzultace

Budou probíhat osobní formou na cvičení a po předchozí domluvě mailem. Cvičící může rovněž v rámci svých časových možností poskytovat konzultace prostřednictvím fóra před-

mětu, které ovšem prioritně slouží pro komunikaci mezi studenty. Dále lze využít fakultní elektronickou poštu.

Konzultace k funkcionálním projektům v letním semestru 2020 poskytuje L. Škarvada.

## 6 Ostatní

Popis v souboru `README`, komentáře v kódu, popisy testů a další texty lze psát česky, slovensky nebo anglicky, avšak konsistentně, tedy jen jedním z těchto jazyků.