Języki i metody programowania 2

Lab 1

1 Zakładamy konto na https://bitbucket.org

1.1 Weryfikujemy e-mail i podajemy nazwę użytkownika (najlepiej kombinacja imienia i nazwiska).

1.2 Zakładamy nowe repozytorium o nazwie *JMP2*. Pozostałe opcje pozostawiamy domyślne.

1.3 Ze strony nowego repozytorium kopiujemy do schowka komendę „*git clone https://*”.

2 Klonujemy repozytorium i wprowadzamy zmiany

2.1 Przechodzimy do terminala systemowego.

2.2 Tworzymy nowy folder na repozytorium i przechodzimy do niego.

2.3 Wklejamy skopiowaną komendę „*git clone …*” i ją wykonujemy. Git clone utworzy nowy folder o nazwie *jmp2* a w nim plik *README.md*

2.4 Przechodzimy do folderu *jmp2*.

2.5 Wykonujemy komendę „*git status*”. Otrzymamy informację że jesteśmy na branchu „master”.

2.6 Tworzymy nowy brach komendą: „*git branch lab/lab1*”

2.7 Listujemy wszystkie branche komendą „*git branch*”. Nowo utworzony branch powinien być widoczny.

2.8 Przepinamy się na nowy branch komendą „*git checkout lab/lab1*”.

2.9 Uruchamiamy aplikację CLion, tworzymy nowy projekt i jako ścieżkę wskazujemy folder repozytorium (jmp2).

2.10 Piszemy kod rozwiązujący zadania

Zadanie 1

Napisz program obliczający silnię. Silnia powinna być obliczana przez funkcję

```
int factorial(int);
```

Zdefiniowaną w katalogu factorial. Wykonaj dwie wersje funkcji:
- rekurencyjną
- iteracyjną

Zadanie 2

Napisz funkcję palindrom, sprawdzającą czy podany jako parametr napis jest palindromem. Funkcja powinna zwracać true gdy napis jest palindromem, a false gdy nie jest. Dodatkowo, napisz proste menu posiadające dwie opcje: *Wyjście* i *Sprawdź palindrom*. Po wybraniu *Sprawdź palindrom* program powinien poprosić o wpisanie słowa a następnie sprawdzić i wyświetlić na ekranie czy podane słowo jest palindromem. Po wybraniu *Wyjście* program powinien kończyć działanie.

Pliki z implementacją: **Palindrome.h/cpp**

Sygnatura metody:

```
bool is_palindrome(std::string str);
```

Pliki nagłówkowe:

```
#include <string>
```

2.11 W konsoli wykonujemy komendę „git status” – nowe pliki będą oznaczone jako „Untracked”.

2.12 Wykonujemy komendę „*git add *.cpp*” – oznaczy ona wszystkie pliki z rozszerzeniem cpp do dodania.

2.13 W konsoli wykonujemy komendę „*git status*” – nowe pliki będą oznaczone jako „Changes to be committed”.

2.14 Konfigurujemy git’a wykonując komendy:
git config --global user.name "your-user-name"
git config --global user.email "your-mail@student.agh.edu.pl"

2.15 Commitujemy zmiany komendą
git commit -m „Silnia i palindrom”

2.16 Zmiany od teraz znajdują się tylko w lokalnym repozytorium. Wrzucamy je na BitBucketa przez wykonanie komendy
git push origin lab/lab1

3 Tworzymy pull requesta

3.1 Przechodzimy do przeglądarki internetowej i do strony *BitBucket -> Settings -> User and group access*

3.2 W okienku *Users* wpisujemy nazwę użytkownika *rafal-pawlik* i klikamy *Add*

3.3 W menu po lewej wybieramy *„Pull requests”*

3.4 W nowo otwartej stronie wybieramy *„Create pull request”*

3.5 Pull request ma być z brancha *lab/lab1* do brancha *master*.

3.6 W okienku Reviewers wpisujemy rafal-pawlik i klikamy „Create pull request”.

Linki:

1. Podstawowe komendy git’a
   https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html