Robimy Kompilator!

Zasady:

- Zadanka można rozwiązywać przy użyciu dowolnego języka programowania
- Za rozwiązanie przy użyciu funkcyjnego lub logicznego języka programowania stawiam słoik prawdziwej Nutelli*
- Nie używamy chatbotów/copilota do programowania
- Zapinamy wrotki
- Należy *spróbować* rozwiązać każde z zadań
- Nie jest konieczne rozwiązanie wszystkich zadań

Celem tych zadań jest bardziej przygotowanie niż kwalifikacja. Poza tym, wasze zgłoszenia pomogą mi lepiej dobrać program do poziomu grupy. Polecam użyć swojego preferowanego języka programowania, bo zamierzam dostosować do tego warsztaty.

Pisać na gmaila: `radrowicki+www21@`.

*Z oferty wyłączeni są prowadzący, studenci informatyki i matematyki, organizatorzy oraz Arkadiusz Wróbel.

Zadanie 1

Napisz program który sprawdza czy zadane *wyrażenie nawiasowe* jest poprawne. Poprawne wyrażenie nawiasowe to dowolny tekst w którym nawiasy się "zgadzają" (nie wiem jak to lepiej wyjaśnić). Przykładowo, `(([]){()})(){{}[]}` jest poprawne, a następujące nie są: `([)]`, `(`, `)`, `(]`. Program ma obsługiwać następujące nawiasy: `()`, `[]`, `{}`; wszystkie inne znaki powinny być ignorowane.

Bonus: jeśli wyrażenie nie jest poprawne, program ma wyjaśnić co jest nie tak. Na przykład: który nawias (tj. na której pozycji w stringu) nie jest zamknięty, które zamknięcie nie pasuje do żadnego otwarcia.

Zadanie 2

Napisz program który zamienia wyrażenia infiksowe na odwrotną notację polską. Wyrażenia korzystają z liczb naturalnych, nawiasów, oraz następujących operatorów: `+`, `-`, `*`, `/`. Przykładowo, wyrażenie infiksowe `2 + 3 * 4` powinno się zmienić w `2 3 4 * +`, natomiast `(2 + 3) * 4` powinno wyprodukować `2 3 + 4 *`.

Program powinien reagować na niepoprawne wejście poprzez wypisanie informacji o błędzie i zamknięcie się z kodem zwrotu `1` (np. przy użyciu czegoś w stylu `exit(1)`).

Zadanie 3

Napisz program który przyjmuje na wejściu wyrażenie w odwrotnej notacji polskiej i oblicza jego wartość. W przypadku dzielenia przez zero, program ma zrobić coś super zabawnego.

Konkluzja

Właśnie napisaliśmy kalkulator

Zadanie 4

Rozszerz kalkulator o zmienne. Przykładowo, wyrażenie `let x = 2 + 2 in x + x` powinno wyliczyć się do `8`. Zmienne powinny być nierekurencyjne: `let x = 3 in let x = x + 1 in x` powinno zwrócić `4`. Ponadto, powinno dać się zagnieżdżać przypisania: `let x = let x = 3 in x + 1 in x` powinno również zwrócić 4. Przypisania powinny dać się nawiasować: `2 + (let x = 1 in x + 2)`. Przypisania powinny wiązać bardzo słabo: `let x = 2 + 2 in x + 2` powinno się nawiasować tak jak `let x = 2 + 2 in (x + 2)` (a nie `(let x = 2 + 2 in x) + 2`).

Zmienne nie mogą zmieniać swoich wartości: “przedefiniowanie” zmiennej nie powinno wpływać na użycia zmiennej poza jej scopem. Przykładowo, `let x = 3 in x + (let x = x - 1 in x) + x` powinno zwrócić `8` (poprzez `3 + (3 - 1) + 3`).