Compilers - Assignment 01

HolyScript

Shah Faisal Khan	**21110156**
Yash Kokane	**20110237**
Aishwarya Omar	**20110008**
Rishabh Patidar	**20110165**

Proposed is the syntax and simple features of the language to be designed. It is called **HolyScript** and files are saved with the extension **.holy**.

- **Program start and end:** Every program script starts with the keyword **Summon** followed by **HolyScript** and ends with the keyword **Doom** (representing end). The code within these keywords is considered for compilation.

  Example:

  Summon HolyScript

  //// code for program

  ///// code ////

  Doom

- **Basic Types: numbers, boolean and strings**

  Basic types like numbers, boolean and strings are assigned using the keywords '**bless**' and '**be**'. Identifiers for these types can have names as strings starting with alphabetical characters.

- **Variables:**
  - Declared using the **bless** keyword, followed by the variable name(identifier) and type in brackets.

    Example:

    bless (int) x = 6;

    bless (char) x = 'c'; or even boolean variables like:

    bless (bool) x = truth;

- **Constants:**
  - Constant needs to be defined using the keyword **be,** followed by the type in the brackets and then the identifier.

    Example:

    be(int) x = 10;

be(char) x = ‘A’;

be(bool) x = truth;

- **Booleans:** Declared using the **bless** or the **be** keyword, followed by true or false values. Important is to note that the true value is denoted by keyword **truth** and the false value is denoted by the keyword **myth.**

  Example:

  bless(bool) x = truth; and **bless(bool) y = myth;**

- **Null values:** Null or empty values are represented by the keyword **Null.**
- **Compound Variables (Arrays):** Arrays are declared as like in C++ using the squared brackets.

  Example:

  bless(int) myarray[] = {1, 2, 3, 4, 5}

  A variable array of the name ‘myarray’ of type int gets created.

- **Indentation, Curly braces and semicolons:** The language does not consider indentation for scope like Python, rather it has lines ending with semicolons and scope is defined using curly braces.

  Example:

```
bless(int) vari = 5;
pledge(vari>0){
        preach(“Glory to the Lord!”);
        vari--;
}
```

- **Conditionals:** Conditionals are structured similar to C++ but the keywords are different. If is referred to by **believe** and else is **else.**

  Example:

```
believe(vari==0){
        preach(“Variable is empty”);
}else{
        preach(“Variabel is not empty”);
}
```

- **Printing to the Window:** Any value can be printed to the window by using the keyword **preach.**

  Example:

  preach(“this is the program”);

- **Loops:**
    - **For:** The keyword **chant** is used to represent **for** loops. The conditions and updation logic is separated by semicolons. Keywords **persist** and **retreat** are used for **continue** and **break** respectively.

      Example:

      ```
      chant(bless(int) i =0; i<8; i++){
              believe(i==6){
                      persist:}
              preach("this is a loop");
      }
      ```
    - **While:** The keyword pledge is used to represent while loops.

      Example:

      ```
      bless(int) x = 10;
      pledge(x>0){
              preach("while loop goes on");
              x- -;
      }
      ```
- **Relational Operators:** <, ==, > for comparisons. Example: **bless(int) x < 5;**
- **Arithmetic Operators:** +, -, *, / and % for addition, subtraction, multiplication, division and quotient respectively.
- **Functions:**

  Functions are defined using the keyword **invoke**, followed by the name of the function and the type of the function definition. The return statement is represented by the keyword **deliver**.

  Example:

  ```
  invoke (int) multiply ( int a, int b){
          deliver (a*b);
  }
  ```
- **Deletion:** Deletion is done by the keyword **sacrifice.**
- **Try - Catch Block:** The try-catch block is represented by the keywords **trial** and **mercy**.

  Example:

  ```
  trial{
          ////code
  ```

```
}
mercy{
    /// code
}
```

- **Switch - Case:** Switch - Case is represented by the keywords oracle and case

  Example:

```
oracle (expression) {
  case value1: ...
  case value2: ...
  default: ...
}
```

- **Exceptions and Error:** Whenever an error occurs the following message will print.

  Error: You have sinned!

- **Full Example:**

  Kadane's Algorithm implementation in HolyScript

```
Summon HolyScript

invoke(longlong)maxSubarraySum(bless(int)arr[],bless(int) n) {
    bless(long long) maxi = LONG_MIN; // maximum sum
    bless(long long) sum = 0;

    chant(bless(int) i = 0; i < n; i++) {
        sum += arr[i];

        believe(sum > maxi) {
            maxi = sum;}

        if (sum < 0) {
            sum = 0;}}
    deliver(maxi);
}
Doom
```