Programming Paradigm Project Task: (Deadline of the submission: 31.08.2023)

Consider an imperative language called 'Tiny'. Tiny contains functions, declarations, definition, arithmetic and logical operations, arrays, if-else conditions, a while loop construct, and operations on arrays similar to C. Tiny contains four primary data types, namely, int, float, char, and bool. The entry point function of Tiny is a function name main and has the fixed declaration "int main()." The function declaration of Tiny is similar to C. Tiny supports *16 bits integers, 32 bits floats, and 8 bits characters*. Tiny does not contain any construct to *input* or *output*. Each variable must be assigned a value after its declaration. All variables must be declared at the *beginning* of the function and before they are used. Global variables are not possible. Some of the sample programs of Tiny are given below:

Valid Programs	Invalid Programs

Program A:
int main () {
bool x;
float y, z;
z = 0;
y = 0;
x = true;
if(x) {
y = y + z;
}
else{
y = y - z;
}
return 0;
}
 	
Program B:
int main () {
int i;
char c;
i = 20;
 	
while (i > 0)
{
	Program A:
	int main () {
	int i;
	i = 20;
	char c; //error: declaration must be at the beginning
	return 0;
	}
	
	Program B:
	int main () {
	int a = 2; //error; declaration should be in next lines
	return 0;
	}
	
	Program C:
	int main () {
	int c;
	c = a; // error a not defined.
	Int a; // not allowed here
	return 0;
	}
	Program D:
	int x; // global not allowed
	int main(){

c = ‘a’;
}
return 0;
}
 	
Program C:
int sum(int a, int b) {
int c;
a = 10;
b = 20;
c = a + b;
return c;
}
 	
int main () {
int a, b, c;
a = 10;
b = 20;
c = sum(a, b);
return 0;
}
 	
Program D:
int main() {
int a[10], i;
i = 10;
while(i > 0) {
a[i] = i;
}
return 0;
}
	x = 3;
	return 0; }

Task:

1. **(20 points)** Write the EBNF style grammar for Tiny **and** Implement a parser for this grammar (should check for valid programs).
2. **(30 points)** Implement a static type system to check **for Overflows (integer and buffer).** The input for your implementation would be a program written in Tiny and the output should dictate **integer overflows or buffer overflows** (if there is

any in the input program). In particular, below is the output for a sample program with an Integer overflow:

Input Tiny Program:

```
int main () {
        int a, b, c;
        a  = 65535;
        b = 2;
        c = a + b;
        return c;
}
```

Expected Output:
The program contains a integer overflow at line 5. The sum a + b overflows the maximum integer size (16 bit integer).

3. Document your design decisions.
4. You are free to use any language of your choice for the implementation. However, the submitted code should be compliable out of the box and contains a **ReadMe** file with instruction on how to run the code. If your code need to install some dependencies, I would strongly suggest to submit a system like Docker. Implementations with compilation problems will be straightforward given zero marks.
5. Plagiarism of any kind is not allowed, i.e., you should not copy from each other, copy from internet etc. We will actively check for plagiarism in the code and if found, you will be disqualified from the course and the exam office will be notified.