ALGONQUIN COLLEGE

ASSIGNMENT SUBMISSION STANDARD AND MARKING GUIDE

***Note 1: About this Document**:*

This document presents the criteria to be used to evaluate your online assignments (A1x, A2x or A3x).

Overview

The Assignment in compilers is progressive. It means that what you are doing in one specific activity, will be used in the next one. Sometimes, activities are related to definitions and documentation. Sometimes, it is related to code, and you are developing parts of a compiler using our model (**Sofia**).

- **About the language**: During this semester, you are supposed to create a language using some basic definitions. The language must be named using a **City name**.
  - For instance, professors will use Sofia – the capital city of Bulgarian (https://en.wikipedia.org/wiki/Sofia) for one reason: it was the city of the original professor in this course, **prof. Svillen Ranev**, whom I had the privilege to work during some semesters.
  - You can imagine any city (for instance, as Brazilian, I would choose “Rio”, etc.).
  - You do not need to have a specific reason, but all teams / individual submissions must consider a different city.
- **About teams**: Teams (only 2 students) are not mandatory, but my experience is that it is really a great opportunity for students to discuss and review their activities.
  - The basic rule is that the team must include students from the same Lab session.
  - Once defined, teams cannot be changed.
  - But it will be possible to compose teams during the course. For instance, if you have worked alone in A11, you can stay with someone else in A12, respecting the rules.
  - Similarly, if at any time, students decide to work alone, there is no problem, but they cannot work with a different partner.

All assignments (A11, A12, A21, A22, A31 and A32) are marked using an appropriate **Marking Sheet** that observes some patterns. The purpose of this document is to give you a detail about these documents and how to be marked. This document is composed by some sections:

- **Part I: Submission Standard:** Where you give general information about what is placed in the assignment submission (usually, a ZIP file containing several documents and the ANSI C source code.

- **Part II: Tasks:** The main activity divided in steps. Some can be related to auxiliary documentation (when necessary) and the code execution with information about your outputs (that should meet with the expected outputs **bit-by-bit**).

***Note 2: About Evaluation Process**:*
*The general proceeding for evaluation is summarized here: you (or when applicable, your team) is always starting with **100% of marks**. However, for each kind of **errors** (for instance, problems in compilation process, output mismatching or missing documentation), you are **losing** correspondent marks. See section about rules and calculus.*

Conventions

- **Assignment Notation:**
  - **Axy** -> represents one of these cases: A11, A12, A21, A22, A31 or A32 (so, x=1..2, y=1..3).
    - *Example: "A21" means that you are performing the Assignment 2 – Task 1.*
    - Basically, Ax1 is more theoretical (a document / a diagram, etc), while Ax2 is related to code development.
- **Notation**
  - [something] -> replace **[something]** with the required information
- **Example**
  - Assignment: [number] → Assignment: A11

Part I - Submissions

- Only one kind of assignment submission is required for this course: **Brightspace digital (BSD)** electronic submission. The contents of BSD submissions will be outlined in the assignment submission section of each individual assignment. It is composed by:
  - **Cover page**: (Documentation file): Where you identify:
    - **Team**: The student identification (individual or 2 students).
    - **File's submission**: When you list the files that you are submitting (it can be a specification – see Assignments Ax1 or code – see Ax2).
  - **For documentation (Ax1):**
    - **Team definition**: Define the names of the students.

- **Solution**: For each task, answer the questions – if It is open, describe your ideas, avoiding problems with plagiarism (mention all references used, if it is) and, especially, think about your original ideas.

➢ **For codes (Ax2):**

- **File Headers**: Where you identify your submission with details.
- **Function headers**: Where you identify each function with specific details.
- **Comments**: Where you emphasize several points of the code that are important for logic and the expected results.
  - **Examples**: Variables, constants, functions, etc.

Note 3: Why standards?
In industry, the code is not an individual property and you need to obey some standards. This requirement is more than merely formalism, but the way you can distinguish how professional you are in following conventions and standards. Remember: being a good programmer is not (more) being able to produce code, but working in teams and contribute to collaborative tasks in development.

1.1. File submission:

❖ Create one **ZIP file** containing all the required files.

➢ For Ax1: Include documents, images or tables (it can be DOC, PDF, PNG, etc.)

➢ For Ax2: Focus on code: all .C and .H files are mandatory

❖ You can submit *several times* before the due date, but only the **latest one** will be evaluated.

➢ Remember that later submissions are allowed for only one week.

Note 4: Progressive Course
In Compilers, all assignments are progressive. It means that previous code from one specific assignment is required for the next. In other words, it is completely required to have a previous "working program" (definition will follow) to continue.

1.2. Cover Page:

❖ All printed assignments must have a cover page with the following information:

➢ **Identification**: [Student Name_1 & ID number_1] / [Student Name_2 & ID number_2]

➢ **Course**: CST 8152 – Compilers, Lab Section: [011, 012, 013]

➢ **Assignment**: [number] (A1, A2 or A3)

- **Professor**: Paulo Sousa
- **Due Date**: [due date]
- **Date**: [submission date]
- **Contents**: [you must list here all the files submitted].

❖ Deductions: Up to 5%.

Note 5: Team / Individual submission
In Compilers, you can work alone or with a partner (no more than 2 people is acceptable). When working in teams, it is required to identify the responsibility of each member in the code (example, by function).
Note that the teams can be the same during all assignments or not – different composition or possibility of working alone is allowed.

1.3. File Headers:

❖ Each of your .h or .c files must have a file header written as ANSI C89 comments and containing the following information:

- **File name:** [my_masterpiece.h or my masterpiece.c, and so on]
- **Compiler:** [MS Visual Studio 20XX, gcc version XX, etc.]
- **Author:** [Student name, ID#]
- **Course:** CST 8152 – Compilers, Lab Section: [011, 012, 013]
- **Assignment:** [number: Ax2].
- **Date:** [the date of the final version of the file]
- **Purpose:** [brief description of the contents].
- **Function list:** [list here all the functions declared/defined in this file. Do not include the function parameter lists and return type, i.e. func(), not int func(int a); the list must follow the order of the function declaration/definition in the file].

❖ Deductions: Up to 5% each.

1.4. Function Headers:

❖ Each of your function definitions must have a function header written as ANSI C89 comments and containing the following information:

- **Function name:** [my_funcXX]
- **Purpose:** [briefly explain the purpose of the function]
- **Author:** [name (your name if you wrote the function)]

- **History/Versions:** [version numbers and dates]
- **Called functions:** [list of function(s) that are called by this function]
- **Parameters:** [for each formal parameter: type, specific range or values if applicable]
- **Return value:** [type, specific values if applicable]
- **Algorithm:** [outline the main steps (sections) only; do not include implementation details; for small and clear functions leave this empty]

❖ Deductions: Up to 10%.

1.5. Comments:

❖ Comment each important line of code.

- Examples: important constants (not necessary for obvious values – Ex: TRUE, FALSE, etc.
- Use your judgment to decide whether the line is important.
  - Important lines: testing some special conditions; dynamic allocation; complex calculations; conversions and so on. Do not comment overly your programs.

❖ Consider that as a programmer, **you need to clarify others** about your code.

- All variable definitions/declarations must be commented.
- The comment must explain, for example, the use of one specific variable, meaning of constants, etc.
- All function segments must be commented (a segment is a sequence of related statement i.e. loops, switches, if-else ladders and sequence of linear statements performing some distinctive task.)

❖ Deductions: Up to 10%.

Part II - Tasks

❖ The main activity is composed by one or **several tasks**.

- You must perform the tests and demonstrate your programming skills using the language (ANSI C), following specifications and performing tests (standard or additional, when required).
- The concept behind here is “working programming”.

Note 6: Working programming
Defined by prof. Svillen Ranev, this concept is essential to get marks in assignments: ***An assignment ANSI C program is a working program if and only if****:*

(1) The program files (including the provided main program and some other .h or .c files) ***compile*** *and link with* ***no errors*** *using the standard project for the test-bed compiler Microsoft Visual Studio (version 19 recommended).*

(2) The program runs as a console application and ***does not crash*** *at run-time when processing the test files.*

(3) The program complies ***following the functional specifications*** *given in the assignment. The functional specifications include names, types, prototypes, and functionality.*

(4) The program ***produces the expected output*** *when processing the input data/files specified in the assignment.*

IMPORTANT NOTE: If output files are provided for the assignment, the program must produce ***byte by byte identical*** *output files when it is run with the corresponding test files. Adjustments to the program, which violate the specifications, and which are implemented with the sole purpose of producing an identical output will render the program non-working.*

Note 7: IDE Environment

The standard project for the test-bed compiler ***Microsoft Visual Studio*** *(version 2019 recommended). However, if you are more comfortable with other IDE's (ex:* ***VS Code****), you can use it, but the project will be tested over MS Visual Studio, which means that eventual compilation problems or warnings can appear in this environment differently and this is the way you will be marked.*

Rules for Calculating Marks – Ax1 (Documentation)

- **Initial Marks**
  - At the beginning you do not have marks, so, you need to answer each question / topic in order to get the marks.
  - Plagiarism will be considered as a fatal error and, when detected, **no marks** will be given at all. So, external references can be used, but they must be cited.
  - **Originality** is the keyword: if you are solving the problems by yourself, at the same time, you do not have risks about plagiarism and, at the same time, you can achieve good marks.

Rules for Calculating Marks – Ax2 (Codes)

- **Initial Marks**
  - At the beginning you receive 100% of the maximal mark for the assignment.
  - And then some deductions might apply.
- **Working Program**
  - If you do not submit a **working program** (see previous definition), you automatically turn into "zero".(0).

- **Warnings Rule**
  - If your program compiles with warnings and those warnings are not justified and not explained in your comments, every kind of warning will cost you 1%.
    - For instance, if 5 warnings from 2 kinds of messages, only 2 marks will be deducted.
- **Late Submission – Factor Reduction (FR)**
  - In order to pass the course and receive a full credit, all assignment must be submitted on time. Due to the specific nature of the course evaluation – deferred success (in other words **F**) if the complete program is not working at the end – late assignment submission will be allowed but there will be significant mark deductions. The deductions for lateness are:
    - **50%** of the assignment allocated marks for a submission up to one week after the due date.
  - **All assignments must be completed to pass the course**, even if the assignments are late. The last assignment cannot be late.
- **Additional marks (Bonus)**
  - These bonuses can be used to help students to get marks even with partial marks (see working program). I might still give you some credits, but your final mark according to:
    - **Minimal deduction**: if small discrepancies (in some files) are found, up to 75% marks can be given.
    - **Mean deduction**: if several files are not matching, with several discrepancies, up to 50% will be deducted.
    - **Major deduction**: if small files are matching, 25% marks is considered.
    - **Full deduction:** No compilation or plagiarism detected: 100% deduction.

- **Plagiarism**
  - The code of the assignment must be originally written by the student submitting the assignment. If it happens that you have "borrowed" the entire code or even parts of the code, you will get a zero (**0**). Some proves of plagiarism include:
    - Old structure and function definition.
    - Corrupted logic (due to old specification).
    - Non-standard specification (coming from previous definitions).

Note 8: New Assignments
I assure you that all assignments are different from previous terms. So, do not fall in temptation of using previous code: beyond committing a mistake, the final code will not work at all.

Final Marks:

- Check the Marking Sheet for each Assignment.

About Demos

- Check the calendar about the days that you need to **demo** your solution during Lab sessions.
- The demos are not related to extra activities, but directly related with assignments.
- In compilers, demos **are mandatory** to get full marks in submission. Otherwise, you are losing marks.
- If you see the CSI organization, basically for one week, you are receiving directions and seeing some examples and in the next week, you / your team is supposed to show your lab.
- Use the demo weeks to get basic feedback, but because the time, maybe only during office hours, you can have better discussion with professors.

Final Messages

Some **inspirational messages** that prof. Svillen Ranev (who taught Compilers for 2 decades at Algonquin College) used to remember:

- *"Experience is the worst teacher; it gives the test before presenting the lesson."* (Vernon Law)
- *"Not everything that counts can be counted and not everything that can be counted counts."* (Albert Einstein)

***Final Note**:*

Use this guide carefully and check if you are respecting it before your assignment submission

Good luck with Assignments!

File update: Aug 30th 2021.