COMP 451

FORMAN CHRISTIAN COLLEGE

(A CHARTERED UNIVERSITY)

COMPILER CONSTRUCTION

Programming Assignment 1

It's an open books and open notes assignment. Use of Internet is allowed. This assignment can be done in groups of MAX two students. You CANNOT share your code with any other group. Any such attempt will be dealt with seriously.

Grading Criteria

Working Code: 80%

Properly formatted Report: 20%

Viva: The grades provided in the assignment depends largely on how you perform in the viva.

A poor performance in viva may result in 60% to 80% drop in your grade. You MUST have a strong understanding of the code and logic.

DONOT share your code, we may use turn-it-in to check for plagiarism.

Copying code from third party or Internet without having a good understanding of the code will result in a ZERO grade in your assignment.

Important: You need to submit a well formatted and well written report for this assignment. The report should carry following sections:

- Introduction about the problem in hand especially well written information about the preprocessor and its functions.
- Detailed and easy to understand description of your logic. Make separate section for each component. In this assignment at least three sections describing each function.
- Start early. **NO additional time in any case what so ever will be granted. You MUST submit the code file (complete or incomplete on or before the deadline). If some one uploads after the due date / time, his / her grades will be capped by 75%.**
- Viva will be conducted for this assignment

Hard Deadline: Code file/s as well as input output files should be submitted on or before 11:49 pm Sunday April 02, 2023 on MOODLE course page.

A hard copy of the report should be submitted on immediately next working day in lab session. For section A this will be Monday April 03, and for section B it will be Tuesday April 04.

Use same format for the assignment report as for the labs.

WE WILL CONDUCT VIVA FOR THE ASSIGNMENT ON THE SAME DAY YOU SUBMIT YOUR REPORT.

Your grade will be much dependent on the way you perform in your viva.

Your submission on MOODLE should carry

- **Three separate code files one for each task along with input file/s. Note that we may check your code using our own input files.**
- **All the output files (Intermediate output files as well as the final output file) for each task.**
- **Your report. Report should be like documentation. Clearly describing your work. Marks for report are based on this.**
- **Make sure that the code as well as the report file must carry the name and roll number of both the group members.**

COMP 451

- **ZIP all the files that you submit and name it with roll number of any one of the group members. Submit this zip file on Moodle. DONOT use rar format.**
- **Each code file MUST carry a data dictionary in the beginning, before your start your code. This should carry following information:**

```
/*
      DATA DICTIONARY
      COMP 451 Section X
      Spring 2023
      Assignment 1
      Group Members: Farhan Ali
                     Omar Farooq
      Roll Number:   123456
                     345678
      Date of Submission: April 02, 2023

*/
Your code should follow from this point onwards
```

COMP 451

Assignment Task-1 [15 Marks]

In this task you will write C code that accepts a file name as command line argument. It checks the validity of the file and opens it in read mode. Program should read the file character by character and should perform the following action in the given sequence:

- Eliminate any blank line/s in the program.
- Identify and eliminate all double slash as well as slash star comments.
- Eliminates unnecessary tabs/spaces in the program.
- You may use intermediate files to write the output of above steps as per your logic.
- Finally, the updated program should be written in a separate file named out.
- Display the file on console.

An input sample and corresponding output is shown for your convenience.

Input file 1

```
/*****in1.c*****/
#include <stdio.h>
//defining macros
#define ON 1
#define OFF 0

void main()
{

      /*declaring variables*/
      int j = 2;
      int    motor,sensorValue =    0   ;
      if(motor   == ON)
      {
            sensorValue++;
      }
      else if(motor ==   OFF)
      {
            sensorVlaue--;
      }
      return 0;
}
```

The out file for this input is:

```
void main()

{

int j = 2;

int motor,sensorValue = 0;

if(motor == 1)

{

sensorValue++;

}

else if(motor == 0)

{
sensorVlaue--;

}

return 0;

}
```

COMP 451

Assignment Task-2 [15 Marks]

In this task we will take up the output file from task-1 and further process it as follows:

File name will be provided as command line argument. Perform appropriate checks on the file and open it in read mode.

We have a file that does not carry any tabs or unnecessary spaces or blank lines. Read the file character by character and write the contents of file in a buffer as a linear array of characters or integers. While writing the contents please note that you should ignore any new line character. All necessary spaces should be included in the buffer. Your program should explicitly place a sentinel value at the end of the buffer carrying linear representation of the program.

Please note that the size of the buffer should be sufficient to hold the entire file.

Your program should write the buffer in an output file.

Make sure your program has the option to display the contents of the output file or input file on console.

A sample input file and corresponding output are shown for your convenience.

<u>Input file (without any comments, unnecessary spaces and blank lines)</u>

```
void main()
{
int j = 2;
int motor,sensorValue = 0;
if(motor == 1)
{
sensorValue = 2;
}
else if(motor == 0)
{
sensorVlaue = 5;
}
return 0;
}
```

Output is as shown below:

```
void main(){int j = 2;int motor,sensorValue = 0;if(motor == 1){sensorValue = 2;}else if(motor ==
0){sensorValue = 5;}return 0;}$
```

Note that the output should be written in a file. This file will be used by task 3 of this assignment.

COMP 451

Assignment Task-3 [50 Marks]

In this task we will take up the output file from task-2 and further process it as follows:

In this task you will implement the algorithm for single buffer scheme discussed in class. In case you like to brush up the concepts please refer to chapter 3 of your text book.

Suppose following is a list of tentative lexemes that may appear in your program.

Identifiers	Keywords	Integers	(	)	{	}	[	]	;	:	#	<	>	=
+	-	*	/	==	>=	<=	!=	All strings in double quote are considered as single lexeme						

An identifier is classified as a word that start with an alphabet and then it can have as many alphabets and digits as needed. Similarly, keywords are also words composed of alphabets. For the time being we are not concerned with the identification of keywords or identifiers.

If your program encounters a double quote then every character until the pairing double quote is found will be assumed as a single lexeme.

Your program should read the buffer (carrying linear representation of the program) and using base pointer bp and forward pointer fp should be able to identify all the lexemes until $ symbol is encountered.

Few sample runs for this task is shown:

Sample Run 1

```
void main(){int j = 2;}$
```

The output of the program should be as follows:

```
Lexeme: void
Lexeme: main
Lexeme: (
Lexeme: )
Lexeme: {
Lexeme: int
Lexeme: j
Lexeme: =
Lexeme: 2
Lexeme: ;
```

Sample Run 2

```
void main(){printf(“Hello Class”);}$
```

The output of the program should be as follows:

```
Lexeme: void
Lexeme: main
Lexeme: (
Lexeme: )
Lexeme: {
Lexeme: printf
Lexeme: (
Lexeme: Hello Class
Lexeme: )
Lexeme: ;
Lexeme: }
```