ASSIGNMENT CODING, SUBMISSION AND GRADING

You are required to submit your code for the assignment to the Automated Programming Assessment System (APAS) for automated marking and grading. Please follow strictly the following guidelines on submission and grading.

- **Submission Deadline** – Submission deadline will be announced for the assignment.

- **Late Submission** – Late submission **will not** be accepted for grading. If you have any problems on submitting the code using the APAS system, please submit your assignment code to our lab staff Ms Eng Hui Fang (ashfeng@ntu.edu.sg) or the lecturer via email before the deadline. Again, late submission after the deadline will not be accepted for grading. As such, the score for that assignment will be 0. Please remember to submit your assignment code before the submission deadline.

- **Grading Results** – Your submitted assignment code will be graded automatically by the APAS system and the results are available for viewing via the APAS system.

- **Request for Review** – For the assignment, after your code has been marked and graded, you may fill up the "Request for Review" form together with supporting evidence on your claims (bs=ased on test cases) if you found that the grading is not correct. Please print the hardcopies of your rquest form and supporting documents, and pass them to Software Project Lab (attn: Ms Eng Hui Fang with email: ashfeng@ntu.edu.sg). We will then review your submitted code manually accordingly. Please note that the period for your request is within one week after your graded assignment is returned. Please also note that **late request** beyond the request period will not be accepted. You may retrieve the form from the course website.

- Please do email the lecturer if you have any queries.

IMPORTANT NOTES ON CODING FOR APAS

Your submitted code for assignment will be graded by APAS (Automated Programming Submission System). As the compiler used in APAS (Linus-based C compiler) is different from that used in Code::Blocks (Windows based C++ like compiler), your program that can run on Code::Blocks may have error when running on APAS. As we only accept submission code that is able to run on APAS, it is important to take note of the following for your coding considerations before you submit your assignment code to APAS for grading:

- Assignment question will provide you with the following:

    (1) problem specification
    (2) program template
    (3) sample input/output sessions with test cases

    These are the problem requirements. **You have to follow exactly the problem requirements to build your programming code for the assignment question.** Failing to do that will result that your code will be treated as wrong answer to the assignment question.

- The grading is carried out based on test cases using program input/output with "**exact string matching**". Therefore, it is extremely important to pay special attention to the program input and output format (including letter cases) when you write your programs (a simple mistake on letter case will cause your program to be marked as incorrect). Please follow exactly the printout format shown in the sample input/output sessions.

- In your coding, please do not change the **main()** function given in the program template when developing your own code. It is used to control the testing cases for checking the correctness of your programming code.

- You do not need to consider error checking on user input unless it is specified explicitly in the problem specification. Just follow the format of the test cases given in the sample input/output sessions.

- Please note that the **sample test cases** are only used for your **testing only**. The data in the sample sessions will be used in the "**pretest**" to tell you which sample test cases have passed or failed. No scores will be given to the sample test cases. As such, they are not used for scoring or grading of your program code. Your program should be designed to be able to run with the different test data.

- In fact, the grading is carried out based on the correctness of the number of **"hidden" test cases** using program input/output using "**exact string matching**". It means it will check the exact input and output of your program according to "hidden" test cases in order to determine whether the program is correct or not.

- We use **"hidden" test cases** (which in fact are similar to those sample test cases) for the grading of your program as this is to ensure that your program code will **not** be **hard-coded** (i.e. **no hard-coding**) to achieve the desired output results, and be able to handle all test data or cases. The scores will be calculated based on the number of correct **"hidden" test cases** that your program can run correctly.

- When you develop your program (your code) in APAS, there are three options available:

  (1) "**Try Compilation**" – It compiles your program and detect compilation errors. Please correct the errors before you can run the code.

  (2) "**Test with Sample Inputs**" - As mentioned earlier, the question requirements will provide you with sample input/output sessions. The "Test with Sample Inputs" option (or **pretest**) will allow you to test your program with the sample input/output cases. It will inform you which sample test cases have failed, if any. You may then rectify your code according to the errors. To do this, please check the test cases from the sample input/output session and modify your code to correct the error.

  (3) "**Run Input**" - The "Run Input" option is very useful as well. You can enter **ALL the program input** in the **Program Input** box, and click the option, then **ALL the output** will be displayed on the **Program Output** box. From there, you may inspect whether your program follows the printout data given in the sample input/output sessions in the question requirements. In case your program has failed for certain sample test cases, you may inspect the printout data and check whether your program follows the printout data exactly or not, and then rectify your program error.

- **Common Errors when Running Your Program in APAS** - As APAS uses a C compiler different from the C++ like compiler in Code::Blocks, please follow the following guidelines when you do your coding:

  - **APAS's C Compiler** – APAS uses a C compiler while Code::Blocks uses a C++ like compiler to compile C programs. Therefore, in order for your code to run correctly in APAS, you need to make sure that your code is written in **C** syntax, **not C++** syntax.

    - **No inline declaration in C -** all **variables** must be declared at the **beginning of the function**. You cannot declare a variable wherever you need to use it. For

example, inline decrlaration such as **for (int i=0; i<size; i++) { .. }** is not acceptable in APAS. It may generate errors.

- **Variable initialization** - Note that you will need to **initialize the values of variables in your functions** (if the initial value is needed for calculation, for example **int total = 0;** will initialize the variable total to 0 before adding more values to it) as different compilers may initialize the local variables with different values. This happens between the compilers running on Code:Blocks and APAS. The initialized values in the two different compilers are different. You have to ensure that your program code is able to run on APAS.

- **Printout Data** - It is extremely important to pay special attention to the **program input and output format** (including letter cases and spaces) when you write your programs (a simple mistake on letter case will cause your program to be marked as incorrect in APAS). Please follow exactly the data printout format shown in the sample input/output sessions. Please remember that it is your responsibility to make sure that your program printout data must follow the data requirements given in the sample input/output sessions.

- "**TIMEOUT**" Error Message in APAS – This message occurs when the program waits for user input and no input occurred. The timeout is set when the program waits for input after a few seconds and if there is still no input from the user, "TIMEOUT" message will be displayed.

- **Reading Data Input from Mixed Data Types** – When you read data input which consists of data from different data types (e.g. to read int first, then read character or vice versa), you may need to consider the extra newline character **'\n'** in the input buffer (this will be discussed in the Chapter on Structures). In APAS, you need to use the statement: **scanf("\n");** to achieve it. The scanf("%c", &dummychar); and getchar() can work correctly most of the time in APAS. **Try NOT** to use the following statement to achieve that as they may not be accepted by APAS:

  - fflush(); will not be accepted by APAS

  The statement will **NOT** be accepted in APAS to help get rid of the extra newline character in the input buffer.

- **Array Processing** – One of the common errors in writing code in array processing is using **index** to access array element that exceeds the size of the array. For example, if size of array is ***array_size***, then the max index will be ***array_size-1***. Some students' code may make a mistake to allow the index **larger than *array_size-1***. As such, this kind of error is called "index out-of-bound". So when you develop your code, please double check that the index you use in your code (after updating the index) does not exceed the size of the array (i.e. *array_size-1*).