

FACULTY OF INFORMATION TECHNOLOGY
31473 – DATA STRUCTURES AND PROCEDURAL PROGRAMMING
SPRING 2007

ASSIGNMENT 1 - DUE 11:59pm, 21/9/2007

This assignment is worth 15% of the total marks for this subject.

Requirement

For this assignment, you are to write a C program that validates International Standard Book Numbers (ISBNs).

Details

International Standard Book Numbers

ISBNs were introduced by publishing houses in the early 1970s to simplify computer processing of orders, inventories, etc. Each ISBN has ten digits. The first nine identify the publisher and the book, while the tenth is used to ensure that the number has been entered correctly, and is called a *check digit*. A similar technique is followed for many other important numbers, such as student numbers and bank account numbers.

Formats

The ISBN has ten digits. These are normally broken into smaller groups with hyphens or spaces to make it easier to transcribe them. The break-up varies, because the publisher definition can be done in several ways, depending on the size of the publisher.

Some examples of ISBN's are:

0-7897-0414-5
 0-262-68053-X
 0 201 54322 2

Notes:

- 1) There is no guarantee that there will always be four numeric groups, so you should consider that the number of characters may be any value of ten or greater.
- 2) The check digit sometimes must have a value of 10, and this is indicated by putting an X in the check-digit location.

Calculation

To check an ISBN, each digit is multiplied by a weight, the products are summed, and the sum must be a multiple of 11. The weights and an example calculation using the first ISBN above are shown in the following table.

<i>Digit</i>	0	7	8	9	7	0	4	1	4	5
<i>Weight</i>	10	9	8	7	6	5	4	3	2	1
<i>Product</i>	0	63	64	63	42	0	16	3	8	5

The sum is 264, which is $24 * 11$, so the remainder is zero and the ISBN is correct.

It is not hard to show that this method will detect any single-digit error, or any transposition of two adjacent digits. These are the most common data-entry errors. About 90% of more complex errors will be detected, but not all: for instance, if one is added to both the fifth and sixth digits, the effect is to add 11 to the sum of products, so this error would not be detected.

Obtaining Input

The user should be able to type the ISBN in the format shown on the book she is checking, with dashes or spaces. The possible presence of spaces in the input means that `scanf` may not process the input correctly. Instead, you should use the function `fgets`, which accepts a whole line of input, including its newline character.

To use `fgets`, you should code as follows (replace fields in *italics* with your actual values):

```
fgets(array name, max_input_size, stdin);
```

An example of its use would be

```
char array[20];  
fgets(array, 20, stdin);
```

For this assignment, we will ignore `fgets`' return value, and signal end-of-input by entering a line that is empty.

As part of this assignment, you may not use an array with `fgets` that is greater than 20 chars in size. If the user enters more than 20 characters on the keyboard then the program must clear any unread characters and then wait for a new ISBN to be entered.

Processing Digits

Your program will accept a line of input containing an ISBN with dashes or spaces, identify the digits, and perform the above calculation. Note that the digits in the input string will be characters, with their associated ASCII value. To convert a single digit character to its corresponding internal integer value for calculation purposes, simply subtract the character code for '0'. E.g.

```
char c = '9';  
int num = c - '0';
```

Will put the value 9 into `num`.

Program Action

Your program should do the following:

- 1) Accept a line of text from standard input.
- 2) Check that the line has a valid ISBN format. If not the program should print an error message and go to step 4). If the line entered is a null line then the program stops.
- 3) Otherwise, it should check the ISBN, print out a statement indicating whether it is correct or not, then wait for the next ISBN.
- 4) Wait for the next ISBN to be entered.

Benchmark Program

To clarify how the program works, a benchmark executable version has been placed on the server. You can run it by typing the following command

```
/home/glingard/isbncheck
```

To give you an idea of the size of this assignment, the source code for the benchmark program, including whitespace and comments, takes about 83 lines of code.

Assignment Objectives

The purpose of this assignment is to demonstrate competence in the following skills.

- Program design
- Array manipulation
- Creating functions in C
- String and character manipulation

These tasks reflect all the subject objectives apart from objective 4.

As part of your subject workload assessment, it is estimated this assignment will take 13 hours to complete.

Queries

If you have a question, please contact the subject coordinator as soon as possible.

Gordon Lingard
glingard@it.uts.edu.au
9514-7935
Room 04.559, Building 10

However, for frequently asked questions a FAQ file will be put up. Please check this file before emailing the coordinator with a question.

```
/pub/prprog/assign/assign1/faq.txt
```

There is already a assignment FAQ in the DS & PP web site. You should read this.

If serious problems are discovered the class will be informed and an update will be included in the following file

```
/pub/prprog/assign/assign1/errata.txt
```

Please keep a look out for this file.

PLEASE NOTE. If the answer to your questions can be found directly in any of the following

- subject outline
- assignment specification
- faq.txt
- errata.txt
- UTS Online discussion board
- DS & PP web page

You will be directed to these locations rather than given a direct answer.

PLEASE NOTE. Please do not send email in HTML format or with attachments. They will not be read or opened. Only emails sent in plain text format will be read. Emails without subject lines will be automatically deleted by the junk mail filters I have in place.

Assignment Submission

You will submit your program via the `submitass1` program. It is assumed that your C file is called `assign1.c` (although you may call it something else).

You must start in a directory that is within your home directory on `charlie` or `sally` and then run

```
/home/glingard/submitass1 assign1.c
```

This assumes that `assign1.c` is in the current directory.

The `submitass1` program will then perform a number of tests. These include.

Compiling

Your program must compile without fatal errors or warnings – using both the `cc` and the `gcc` compilers. Your program will be compiled with the following commands and options

```
cc
gcc -pedantic-errors -Wall -Werror
    -Wmissing-declarations -ansi
```

You can learn what these compiler options mean by running `man cc` and `man gcc`.

PLEASE NOTE. Your program must compile on the student UNIX server. Programs written on Window's machines sometimes don't compile or run properly on the student server.

Style Feedback Program

Your program will be run through a style feedback program, which will check that your code meets a minimum style layout. If your program does not meet the standard a warning message will be printed. The style feedback program will display messages showing which lines of code do not conform to the standard and why they don't.

Code Feedback Program

Your program will be run through a code feedback program, which will check that your code meets minimum coding practices. If your program does not meet the standard a warning message will be printed. The code feedback program will display messages showing which lines of code are not acceptable and why they aren't.

Test Filter

Your program will be tested with 4 different sets of tests. The results of your program will be compared to the results generated using the benchmark `isbncheck` program.

PLEASE NOTE. One of the tests that will be applied to your program will involve a line of text of more than 20 characters. Students who increase the array size used in the `fgets` call to deal with this problem will receive 0 for the test.

PLEASE NOTE. Make sure the output from your program is ***EXACTLY*** the same as that from the benchmark executable. ***ANY*** deviation of the output from your program to that of the benchmark executable will cause the test to fail.

Plagiarism Agreement

You will be required to agree to a statement that the assignment is your own work and that you haven't given your code to anyone else.

If all goes well, `submitass1` will reply with a message saying you have successfully submitted the assignment. It will also place in your current directory a file called `receipt.txt`.

You may submit your assignment as many times as you like. The last assignment received will be the one marked.

PLEASE NOTE. Only assignments submitted via the `submitass1` program will be accepted for marking.

receipt.txt

`receipt.txt` is your proof that you have submitted your assignment. Once you have received it copy it to somewhere safe, such as your home directory. Additionally, copy your C file into the same location. Do not modify them in any way. If you wish to continue developing your program then do it on a duplicate file.

The `receipt.txt` file contains three pieces of information

1. A line saying you have submitted your file and when you did it.
2. A checksum of your C file
3. A checksum of your receipt

If you tamper with your C file or the receipt then the checksums will be become invalid and therefore your receipt will become invalid. No actions will be taken if receipts have invalid checksums.

Acceptable Practice vs Academic Malpractice

- ❑ Students should be aware that there is no group work within this subject. All work must be individual. However, it is considered acceptable practice to adapt code examples found in the lecture notes, labs and the text book for the assignment. Code adapted from any other source, particularly the Internet and other student assignments, will be considered academic malpractice. The point of the assignment is to demonstrate your understanding of the subject material covered. It's not about being able to find solutions on the Internet. You should also note that assignment submissions will be checked using software that detects similarities between students programs.
- ❑ Do not let anyone submit their assignment from your account. The `submitass1` program copies your assignment into a secure directory based upon your user login name. Anyone else using your account will have their assignment placed in your directory. Students who do this will find themselves reported to the Faculty for possible academic malpractice and misuse of Faculty resources.
- ❑ Participants are reminded of the principles laid down in the "Statement of Good Practice and Ethics in Informal Assessment" in the Faculty Handbook. Assignments in this subject should be your own original work. Any collaboration with another participant should be limited to those matters described in the "Acceptable Behaviour" section. Any infringement by a participant will be considered a breach of discipline and will be dealt

with in accordance with the Rules and By-Laws the University. The Faculty penalty for serial misconduct of this nature is zero marks for the subject. For more information, see http://wiki.it.uts.edu.au/start/Academic_Integrity

Assignment Security

It is important to note that you have a responsibility to maintain the security of your assignment files. You can read more details about this in the Academic Malpractice section of the DS & PP web site - <http://learn.it.uts.edu.au/prprog/>

Assessment

Marks will be awarded out of 20 based upon the following scheme.

- Between 0 and 8 marks will be awarded by the computer based on the number of tests passed. 2 marks for each test passed.
PLEASE NOTE. Some students have been known to write their code to artificially pass the tests rather than solve the assignment problem. In such cases a reduced mark will be given for the tests, including being given a 0.

The following marks will only be awarded if a score of 6 or more is awarded for the tests and both the code feedback and the style feedback programs are passed without generating warnings.

- Between 1 and 9 marks will be awarded on the quality of your code design and the algorithms used. The appendix in the lecture notes has a section called *Program Design* for further clarification on this.
- Between 1 and 3 marks will be awarded on the presentation style of the program. This involves meaningful variable names, intelligent use of comments and so forth. The appendix in the lecture notes has a section called *C With Style* for further clarification on this.

For further details of the marking scheme please check out the assignments section of the DS & PP web page. <http://learn.it.uts.edu.au/prprog/>

Late Assignments, Extensions and Special Consideration

Please read the subject outline regarding late assignments and extensions. If you did not get the outline a copy can be found at the DS & PP web site

<http://learn.it.uts.edu.au/prprog/>

Assignments that are late by less than one week will incur a penalty of 1 mark for each day or part thereof late. Assignments more than a week late will not be accepted under any circumstance as the assignment solution will have been made available by then.

An extension of one week will only be granted if there is a fully documented reason which merits it. The documentation must be presented to the Subject Coordinator before the assignment due date. Extensions longer than a week will not be granted under any circumstance. If a one week extension is granted that means the assignment will be accepted up to a week after the due date without penalty. It will not be accepted later than that.

Students may apply for special consideration if they consider that illness or misadventure has adversely affected their performance in the assignment. For more information go to

http://www.sau.uts.edu.au/exams_ass/spec_cons.html

Return of Assessed Assignments

The code of your assignment will be printed out and marked. It will list codes for particular types of programming issues. These codes can be found in the DS & PP web site in the `assessment->assignment->assessment` menu section and will give a complete break down of your marks.

The marked assignments will be returned via the Student Center on level 2 of Building 10. The estimated return date is 5/10/07.

Getting Marks

You can check on your marks by running the following program.

```
/home/glingard/getmarks
```

Apart from your marks this program will give the following information.

- Given out – the date the assignment is given out.
- Due date – the date the assignment is due.
- Late date – assignments will be accepted up to one week after the due date but with a penalty of -1 mark per day late.
- Marks released – The date the marks are released.
- Close date – The assignment is closed and the solution will be released. The close date will be one week after the assignments are marked and given back.

PLEASE NOTE. It is your responsibility to check that I have received your assignment and given you a mark. Even if you have a receipt you should check your mark and inform me if there is any problem before the close date. The `getmarks` program allows you to do this very easily. Unless you have a valid receipt or there are exceptional circumstances (e.g. serious medical conditions) no further correspondence regarding the assignment will be entered into after the close date.