**CPE100 International Sections, August 2020**
**Programming Project 1 – Input Validation**
***Updated: 30 September 2020***

### *Assignment Objective*

The objective of this assignment is to give you a chance to design, write and test a fairly large program. This program will use the concepts we have been studying, including conditionals, looping, terminal input and output, arrays, and functions. It will also give you the opportunity to develop code which will be useful in Project 2. Finally, you will get the chance to practice using good programming style.

### *Assignment Description*

One common task in programming is to *validate* input. "Validate" means to check some data to make sure it follows rules about its content and format. For instance, you might want to validate an exam score entered by the user, to make sure it is a number between 0 and 100.

A validation function typically receives the data to be checked as an argument, and returns true if the data are correct ("valid"), and false if some error is found. A validation function is more useful if, in the case of invalid input, it also provides some information about the type of error it found (for example, printing a detailed error message).

For CPE100 Project 1, you will create a program that includes four different kinds of validation. In the main function of your program, you will display a menu, asking the user what they want to do (which of the four validation tasks). After the user chooses a validation task, your program will loop, asking for input for this validation, and printing the result, until the user hits the <Enter> key without typing any text. Then the program will go back to the main menu.

At the end, your program must print a summary listing the total number of validations it performed, for each type, for the full program run.

For example, a sample program run might look like this:

---

```
./validate
Validation options:
    1 – Check date in form dd/mm/yyyy
    2 – Check Thai mobile phone number
    3 – Check email address
    4 – Check time in 24 hour format
    5 – Exit the program
What do you want to do? 2

Validate Thai Phone Number (Hit return to stop)
    Enter phone number: 09828
        Not valid – number is too short
    Enter phone number: 2318888888
        Not valid – number must begin with 0
    Enter phone number: 07abcd121111
        Not valid – numbers only
    Enter phone number: 0982226766
        Valid
    Enter phone number: 0600000000
        Valid
    Enter phone number:


Validation options:
    1 – Check date in form dd/mm/yyyy
    2 – Check Thai mobile phone number
    3 – Check email address
    4 – Check time in 24 hour format
    5 – Exit the program
What do you want to do? 1

Validate Date dd/mm/yyyy (Hit return to stop)
    Enter date: 22/10/2015
        Valid
    Enter date:
```

```
Validation options:
    1 – Check date in form dd/mm/yyyy
    2 – Check Thai mobile phone number
    3 – Check email address
    4 – Check time in 24 hour format
    5 – Exit the program
What do you want to do? 2

Validate Thai Phone Number (Hit return to stop)
    Enter phone number: 555
        Not valid – number is too short
    Enter phone number:

Validation options:
    1 – Check date in form dd/mm/yyyy
    2 – Check Thai mobile phone number
    3 – Check email address
    4 – Check time in 24 hour format
    5 – Exit the program
What do you want to do? 5

Program run summary:
    Validation type 1:  1
    Validation type 2:  6
    Validation type 3:  0
    Validation type 4:  0

Goodbye!
```

---

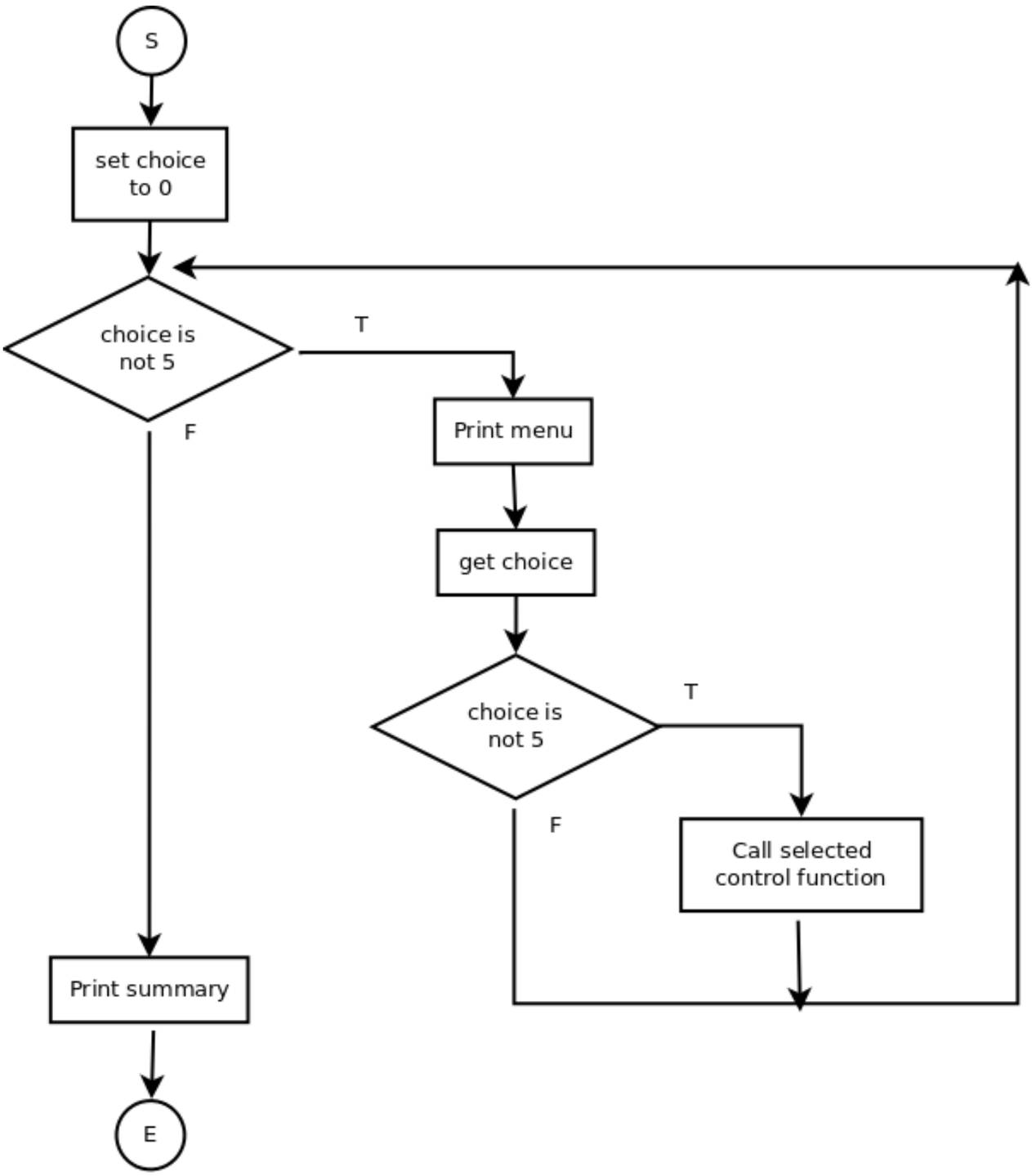
As you can probably tell from the example above, the program will have two levels of loop. There is a loop in the main function, for choosing the type of validation. There will also be a loop in each validation branch, repeating to get new data to validate.

The program flow is shown in the two flowcharts on the next two pages.

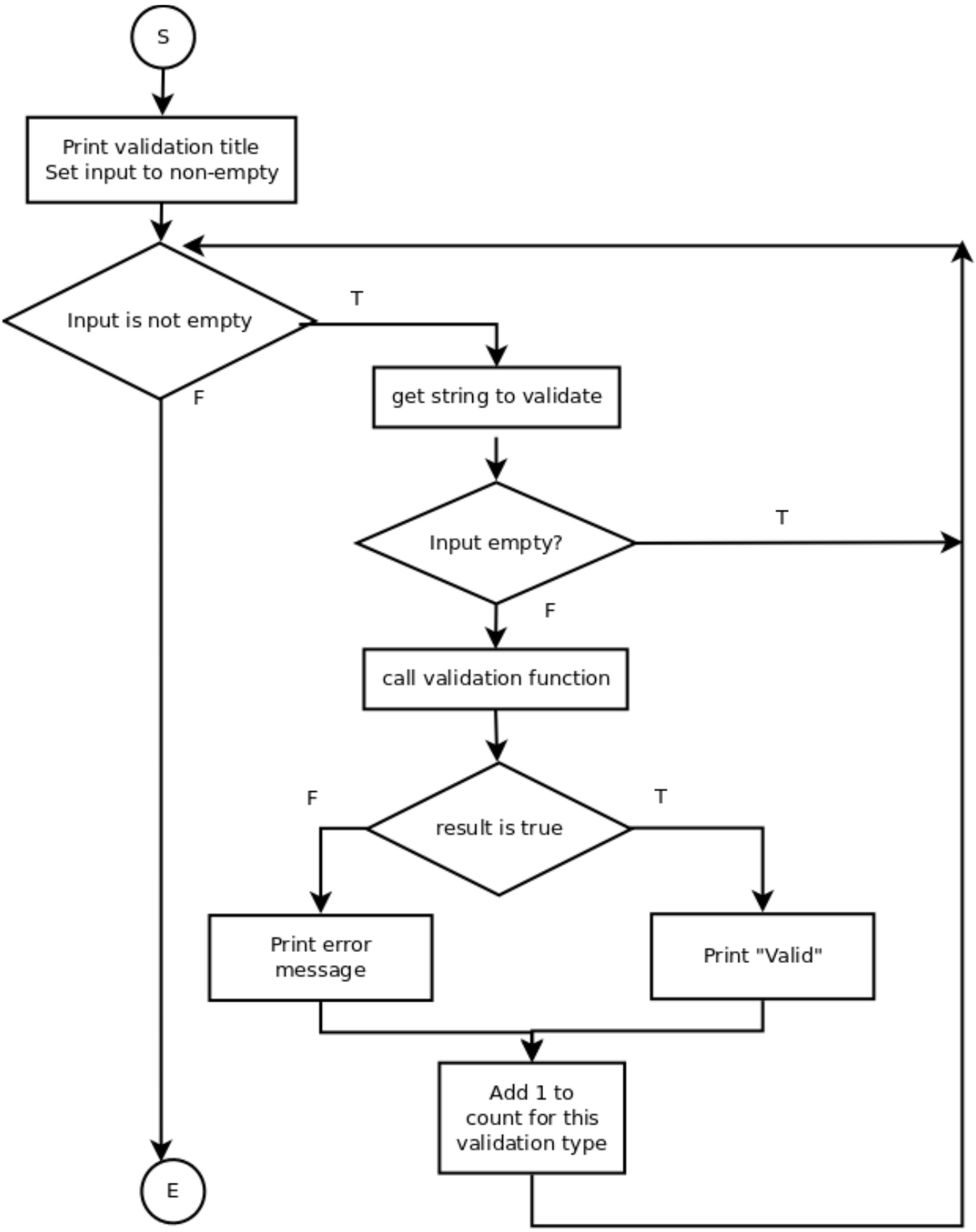
Your program **must** have at least three levels of functions:

1. Main function to show the menu, process user choices and display summary;
2. Control function for each type of validation, which will repeatedly ask for input to validate;
3. Validation functions, which will check a single input and return the results. These functions can also print error messages, or you can return some information to the control function to let the control function do this.

You may want to create other functions as well, to help simplify your code. Remember that our coding standards say a ***function should never be more than one page long***. “One page” is about fifty lines.




**Flowchart for main function**




**Flowchart for control functions**

There are seventeen different validation problems, which you will find at the end of this document. Each student will be assigned a different set of problems. You will find your assigned problems here:

http://windu.cpe.kmutt.ac.th/cpe100/Project1Assignments.html

You ***must*** implement validation for the problems you have been assigned. You should list them in your menu ***in the same order*** as they are listed in the assignment page above (but number them 1 through 4 in your selection menu). For instance, if you are assigned problems 2, 12, 5 and 9, your main menu should look like this:

```
Validation options:
    1 – Validate date in form dd-mm-yyyy – Western year
    2 – Validate CPE student ID number
    3 – Validate email address - simplified
    4 – Validate passport number with country code
    5 – Exit the program
What do you want to do?
```

The order is important because I will grade the correctness of your answers based on an automatic script that ***assumes*** you have followed instructions. If your problems are listed in the wrong order, your answers will be counted as wrong – even if your code is correct! So for the sake of your grades, please ***follow instructions***!

***What You Need to Submit***

Each student must submit a **booklet in PDF form** that includes the following:

- A ***cover page*** with your name, nickname, student ID, and ***a list of your four assigned problems*** (numbers and titles)
- One ***flowchart***, or one ***page of pseudocode*** for each of the four assigned validation problems. This should show the ***logic*** you will use to validate one input – it should not duplicate your code. For instance, it should not use the names of variables or use C constructions like array references. You should create the flowcharts ***BEFORE*** you write the code, to help you understand the problem.
- Screen captures that show your ***testing runs***. You should show separate test runs for each of your four validation problems, including both valid and invalid input.

**You will upload the booklet and your C source code to Windu**. I will provide a special link for uploading your PDF file and code.

This assignment is due **at noon on Friday October 23rd.** Late submissions will not be accepted (that is, if you submit late, you get a zero score).

***Grading***

This assignment is worth 15 points of your course grade. Your project grade will be based on the following:

- Maximum 6/15 points based on the ***correctness and completeness of your design*** (your flowcharts or pseudocode). I will also consider how well you have broken up the program functionality into functions.
- Maximum 6/15 points based on the ***correctness of your code and your testing***. Your code should ***satisfy the requirements***, that is, it should behave as described in the detailed problem descriptions below. We will also grade you on whether your test cases do a good job of exercising the program and demonstrating its correctness.
- Maximum 3/15 points based on ***programming style***. You must follow the detailed coding standards that we have presented during the labs, regarding variable naming and capitalization, indenting, comments, use of braces, use of *#define*d symbols, and so on. Having good comments for each function is especially important.

A copy of the coding standards can be found on the course web site.

**This is an individual project.** Each student must submit his or her own work. In the case that we think two or more students have copied from each other, all students involved ***will get a zero on the project*** (15 points off the course grade) ***plus have an additional 10 points*** deducted from their course grade for cheating, as described in the course syllabus. Basically, if you cheat, you are likely to fail the course.

***Validation Problems***

You have been assigned to implement four of the seventeen problems below. Be sure to read the details carefully. If you have questions, ask me ***by email, on the FB group, or in class***. I are happy to help.

You ***must*** implement the specific problems assigned. If you do other problems, I will assume that you copied your work from someone else and will apply the cheating policy described above.

The problem descriptions below include some sample valid and invalid data. You should not assume that the invalid data covers all possible error cases. There are many more ways to be wrong than to be right!

**1. Validate date in form dd/mm/yyyy – Buddhist year**
The 'dd' means a two digit day. The 'mm' means a two digit month. The 'yyyy' must be a year in Buddhist year format. Your function must check that the string entered by the user has the correct format. You must also check that that the date is legal. Years can be no more than 100 years in the past or 100 years in the future.

| **Valid Input** | **Invalid Input** |
|---|---|
| 12/10/2561 | 12/aa/2560 (alphabetic characters) |
| 03/05/2463 | 31/06/2560 (illegal day of month) |
| 11/11/2663 | 12/10/2017 (not BE year) |
| | 29/02/2554 (not a leap year) |
| | 9/8/2449 (missing leading zeros) |
| | 09-12-2548 (wrong delimiter – not /) |
| | 08/09/2770 (outside allowed year range) |

**2. Validate date in form dd-mm-yyyy – Western year**
The 'dd' means a two digit day. The 'mm' means a two digit month. The 'yyyy' must be a year in Western year format. Your function must check that the string entered by the user has the correct format. You must also check that that the date is legal. Years can be no more than 100 years in the past or 100 years in the future.

| **Valid Input** | **Invalid Input** |
|---|---|
| 12-10-1987 | 12-aa-1987 (alphabetic) |
| 03-05-2025 | 31-06-2018 (illegal day of month) |
| 11-11-1920 | 12-10-2560 (BE62070503433 not Western – or year too big) |
| | 29-02-2001 (not a leap year) |
| | 9-8-1999 (missing leading zeros) |
| | 09/12/2018 (wrong delimiter, should be /) |
| | 22/10/1810 (year is outside 100 year range) |

**3. Validate date and time in form yyyy-mm-dd hh:tt - Western year**
The 'dd' means a two digit day. The 'mm' means a two digit month. The 'yyyy' must be a year in Western year format. The "hh:tt" is a time in 24 hour format, where "h" is an hour and "t" is a minute. Your function must check that the string entered by the user has the correct format. You must also check that that the date and time are legal. Years can be no more than 100 years in the past or 100 years in the future. The time part is required. Hours cannot be more than 23. Minutes cannot be more than 59. There must be exactly one space between the day and the hour.

| **Valid Input** | **Invalid Input** |
|---|---|
| 1987-12-10 18:20 | 1987-12-aa 18:20 (alphabetic characters) |
| 2025-03-05 09:04 | 2018-06-30 25:12 (illegal hour) |
| 1920-11-11 23:59 | 1970-05-34 12:30 (illegal day) |
| | 2001-13-12 09:22 (illegal month) |
| | 09-08-1999 16:04 (wrong order of fields) |
| | 09/12/2018 (wrong delimiter, no time) |
| | 1919-11-11 20:52 (year outside range) |

**4. Validate date in form dd MMM yyyy - Western year**
'MMM' is first 3 chars of month, capital letters. As above, you must make sure that the date is both formatted correctly and a legal date. Years cannot be more than 100 years in the past or the future.

| **Valid Input** | **Invalid Input** |
|---|---|
| 12 DEC 1987 | 31 JUN 2055 (invalid day of month) |
| 05 MAY 2015 | 22 AAA 2018 (invalid month) |
| 31 AUG 2100 | 05 SEP 1919 (year outside allowed range) |
| 01 JUN 1920 | 16 SEPT 2003 (i62070503433nvalid month abbreviation) |

**5. Validate email address - simplified**
It is very difficult to write a function that will validate any legal email address. For this project, your function should check the following rules:

- No embedded spaces or special characters anywhere in the email address, except period (.), at-sign (@), underscore (_) and/or dash (-)
- Must begin with an alphanumeric (letter or digit) character
- Exactly one occurrence of the at-sign (@)
- At least one alphabetic or numeric character before the at-sign
- Address must end in one of: **.com, .net, .ac.th, or .co.th** (top-level domains or TLD)
- At least one alphanumeric character between the at-sign and the TLD.
- No underscores after the at-sign.
- At least one alphanumeric character between any periods that occur after the at-sign.

| **Valid Input** | **Invalid Input** |
|---|---|
| a@b.com | jacky$@hotmail.com ($ is not allowed) |
| seg@goldin-rudahl.com | _@mail.kmutt.ac.th (must begin with letter or number) |
| mary_preston@mail.kmutt.ac.th | fred@dummy.zqz (invalid TLD) |
| helen..jones@myemailprovider.net | joe@..com (two periods together after the @ sign) |
| 22up@mars.lander.net | jill@myfavoritestore@johnson.net (two @ signs) |
| 3@my.email-domain.co.th | 3@my_email_domain.co.th (underscores after @) |

**6. Validate Thai mobile phone number**
A valid mobile number must follow these rules:

- Only digits and dashes allowed (no blanks)
- Must include ten digits
- First two digits must be: 01, 05, 06, 08 or 09.
- Up to two dashes allowed (but ***not*** required)
- First or second character must not be dash
- Cannot have two dashes together
- Last character must not be dash

| **Valid Input** | **Invalid Input** |
|---|---|
| 0834445670 | 098111 (too short) |
| 06-3455-7889 | 34-09820111 (invalid prefix) |
| 05-12101010 | 0-607878888 (dash in wrong place) |
| 091-999-8988 | 0512--898099 (two dashes together) |
| 09-6-2321011 | 083a2376771 (alphabetic) |
| 0832-712119 | 083-9098812- (final dash) |

**7. Validate international phone number**
A valid international phone number must follow these rules:

- Only digits, dashes and one plus sign (+) allowed (no spaces)
- Must begin with a plus
- After the plus, must have between 1 and 3 digits for the country code
- After the country code, must have a dash
- After the dash, must have between 1 and 3 digits for a city/area code
- After the area code, must have a dash
- After the second dash, must be between 6 and 10 additional digits.

You do **not** need to check that the country code is the code for a real country.

| **Valid Input** | **Invalid Input** |
|---|---|
| +66-2-23212717 | 66-2-23212717 (no plus) |
| +86-121-7777888898 | +1-213-a787777 (alphabetic character) |
| +1-213-2310101 | +66-29099999 (missing dash) |
| +355-22-768818 | +1(413)2342211 (parentheses not allowed) |
| | +23-78-555 (too short) |
| | +55-7654-988888 (city/area code too long) |

**8. Validate first and last name with title**
The user must enter exactly three strings separated by spaces: title, first name, last name. No special characters are allowed except a period after the title. Upper or lower case are both accepted for any part of the name and title. First and last names must be at least 2 characters long and no longer than 30 characters (each). Title must be one of: **Mr., Ms., Miss, Mrs., Dr.** or **Ajarn** Except for the titles "Miss" and "Ajarn", the terminating period is required.

| **Valid Input** | **Invalid Input** |
|---|---|
| Mr. Jack Jones | Robert Redford (no title) |
| Dr. Sally Goldin | Dr. Sally (no last name) |
| Miss AA jj | Ms. A B (first and last names too short) |
| Ajarn sa ABCDEFGHIJKLMNOPQRSTUVWXYZ | Mr Fred Jones (no period after title) |
| miss anna oakley | Madam Jennifer Jackson (invalid title) |

**9. Validate passport number with country code**
The passport number must begin with a valid two letter country code, as specified here: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements

After the country code, the passport must have between 8 and 14 alphanumeric characters.

| **Valid Input** | **Invalid Input** |
|---|---|
| TH-2219991991 | 121222221 (no country) |
| CN-W19299A9999BA | XZ-37377377779 (invalid country) |
| ZA-ABCDEF22 | ZA-23456 (too short) |
| US-504402111 | US 767272727 (wrong format – space not dash) |

**10. Validate hotel services code**
Suppose we have a website that provides information about hotel services. The site uses a code made up of the following letters:

- P – means the hotel has parking
- R – means the hotel has a restaurant
- W – means free WIFI available
- S – means it has a swimming pool
- T – means airport transportation is available.

A hotel services code is a set of these letters, with no embedded spaces spaces, followed by 1 to 4 dollar signs ($) indicating how expensive the hotel is. The letters in the code can be in any order, but cannot be repeated. The letters must be in upper case. It is possible to have no letters, just the price indicator.

| **Valid Input** | **Invalid Input** |
|---|---|
| TWP$$ | W (no dollar sign) |
| SRPTW$$$$ | TWTP$$ (T is repeated) |
| $$ | SKP$$$ (K is not an allowed character) |
| P$$$ | S W T$ (embedded spaces) |
| | wrs$ (lower case) |

**11. Validate bank account number**
A bank account number is exactly 11 digits, with or without dashes. If there are dashes, they must be in the fourth and tenth positions, that is, like this: *ddd-ddddd-ddd*. The first three digits indicate the branch. They must be one of the following: **206, 209, 128, 921, 403, 421**. The last three digits must include at least one zero.

| **Valid Input** | **Invalid Input** |
|---|---|
| 209-18772-002 | 444-45678-098 (wrong prefix) |
| 12845189905 | 206-78888908 (missing dash) |
| 921-00000-000 | 209-18722-778 (no zero in the ending triplet) |
| | 921-4324-087 (too few digits) |
| | 209-A3455-000 (not all digits) |

**12. Validate CPE student ID number**
The ID numbers used in CPE for the past eight years are 11 digits long. They have the following form: *yy07050ppdd*.

The 'yy' is the last two digits of the B.E. year in which the student entered KMUTT, for example, 55 would mean the student started in 2555. The 'pp' depends on whether the student is an International (34) or Thai program (10) student. The 'dd' can be any two digits except '00'.

Your validation function should accept as valid any student ID number for years from 2555 to 2563 that follows this pattern. Any other pattern should be an error.

| **Valid Input** | **Invalid Input** |
|---|---|
| 56070503412 | 54070503412 (invalid year) |
| 60070501023 | 5607053412 (too short) |
| 59070501001 | 59660503457 (wrong middle digits) |
| 61070503499 | 57070503400 (ends in 0) |
| 62070501001 | 590705A3422 (contains alphabetic character) |
| 63070503499 | 6307 503499 (embedded space) |

**13. Validate street address in Bangkok (simplified)**
Your function should check an address as follows:

- Begins with a number (the house number). The number can include a single slash, e.g. "34/12", but must have at least one digit before and one digit after the slash.
- Next a street name which can include numbers as well as letters;
- Next an *optional* label "Road", "Street" or "Lane";
- Finally, a postal code, which must be five digits and which must begin with "10".

| **Valid Input** | **Invalid Input** |
|---|---|
| 2 Sukhumvit Road 10110 | Sukhumvit Road 10110 (no house number) |
| 43/219 Thonglor 10234 | 43a Mountain Street 10211 (alphabet in number) |
| 150 16 Lane 10322 | 12 Rama2 Road 1022 (invalid postal code) |
| | 45/120 Prachauthit Road 13111 (invalid postal code) |
| | /120 Prachauthit Road 10140 (number starts with /) |
| | 45/ Prachauthit Road 10140 (number starts with /) |
| | 4/3/11 RamaI Road 10119 (two slashes) |
| | AB Soi16 10110 (alphabet in the number) |

**14. Validate IP Address**
IP (Internet Protocol) addresses are the way that the Internet identifies individual computers. An IP address is a series of four numbers which can range from 0 to 255, separated by periods (.), in the form *ddd.ddd.ddd.ddd*

IP addresses do not require leading zeros (e.g. 023) in the four numbers but this is legal. Some of the numbers can be zero, but the first one cannot be zero, and all four cannot be zero.

| **Valid Input** | **Invalid Input** |
|---|---|
| 202.44.12.16 | 300.10.02.10 (> 255) |
| 1.5.003.20 | 0.10.200.90 (first is zero) |
| 2.2.0.0 | 202.44.33.12.98 (too many parts) |
| 55.126.200.255 | 0.0.0.0 (all zeros) |
| | 23.aa.170.23 (alphabetic) |
| | 22..43.012.23 (double period) |

**15. Validate web URL (simplified)**
Validate a URL string according to the following rules:

- Must start with either **http://** or **https://**
- Must end in **.com**, **.net**, **.co.th**, **.ac.th**, or **.go.th**
- Aside from the above, can contain letters, numbers, dashes (-), and periods (.)
- Must not have a period or a dash immediately after the prefix (http:// etc.)
- Must not contain two periods or two dashes next to each other.

| **Valid Input** | **Invalid Input** |
|---|---|
| https://www.flashaction.co.th | www.flashaction.co.th (no prefix) |
| http://windu.cpe.kmutt.ac.th | http:myblog.com (wrong prefix) |
| https://google.com | https://neverland.biz (wrong ending) |
| http://www.center-court.com | http://friendly..skies.net (two periods) |
| https://my.friendly-skies.co.th | https://.action.adventure.com (period after prefix) |
| | https://-your.favorite.store.net (dash after prefix) |

**16. Validate a string as a C language identifier**
In Lecture 2 we studied the rules for valid C identifiers. Your function should correctly distinguish legal from illegal C identifiers. (This validation should NOT consider our coding standard rules.)

| **Valid Input** | **Invalid Input** |
|---|---|
| totalStudents | total Students (embedded space) |
| __MAXVALUE | 2ndString (starts with number) |
| the_last2 | the-last2 (dash is not legal) |
| ItemCode | price$ (dollar sign is not legal) |

**17. Validate password against a set of rules**
Write a function that will correctly determine whether a password string follows these rules:

- At least 8 characters long and no longer than 12 characters
- Must contain at least one upper case letter
- Must contain at least one lower case letter
- Must contain at least two digits
- Must contain at least one of the following special characters: **? @ % $ #**
- Must not contain any other special characters not in the list above.

| **Valid Input** | **Invalid Input** |
|---|---|
| aPassword12? | MyPassword12?? (too long) |
| $$22AbcD | 2ndString6 (no special character) |
| ?@XxXx789 | A 33Space? (embedded space) |
| Sally13%# | alllower55? (no upper case) |
| | aPassword12! (invalid special character) |
| | aA22% (too short) |