SC1003
Review Lecture
Week 8

Review Lecture – Week 8

- **Course Introduction**
- Week 8 – Learning Materials
  - Lectures
  - Lab and Tutorial
  - Code::Blocks IDE
  - LAMS MCQ Questions
  - Coding Practice Questions
  - APAS
- Reviews on Basic C Programming and Control Flow
- Examples

Course Information on C Programming

- Dr. Hui Siu Cheung
- Tel: 6790-4930
- Email: asschui@ntu.edu.sg

- Learning Activities/Schedules are available in NTULearn
- E-Learning course – lecture videos
- Review Lectures
  - 1 lecture hour/week (start from **week 8**)
- Lab-Tutorials
  - 1 lab hour + 1 tutorial hour/week (start from **week 8**)

Course Assessment

- Continuous Assessment
  - Weeks 1-7 (Python Programming) – 50% of the whole course on assessment
  - Weeks 8-13 (C Programming) - 50% of the whole course on assessment:
- Weeks 8-13 (for C Programming)
  - Assignment (35% / 100%)
  - MCQ Test (35% / 100%)
  - Coding Test (30% / 100%)

Learning Schedule (Week 8 – Week 13)

Week	Week 8 4 Oct	Week 9 11 Oct	Week 10 18 Oct	Week 11 25 Oct	Week 12 1 Nov	Week 13 8 Nov	Week 14 15 Nov
Topics	**Basic C Programming and Control Flow**	**Functions and Pointers**	**Arrays**	**Character Strings**	**Structures**		
Review Lecture	Date: **4 Oct 2021 (Monday)** Time: 9:30am-10:30am Online: **MS Teams** (the link for the online lecture is given at the end of the table)	Date: **11 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **18 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **25 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **1 Nov 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)		**Lab Test (MCQ Test & Coding Test)** **Dates: 15 Nov (Mon) and 16 Nov (Tue)** **Details will be announced when confirmed.**
e-Learning Lectures	Learn: Course Introduction Learn: (1) Basic C Programming; (2) Control Flow	Learn: (1) Functions and (2) Pointers	Learn: (1) 1-D Arrays and (2) 2-D Arrays	Learn: Character Strings	Learn: Structures		
Lab-Tutorial	Learn: CodeBlocks IDE Do: Lab-Tutorial 1 (Qns are also available in APAS>Exercise)	Do: Lab-Tutorial 2 (Qns are also available in APAS)	Do: Lab-Tutorial 3 (Qns are also available in APAS)	Do: Lab-Tutorial 4 (Qns are also available in APAS)	Do: Lab-Tutorial 5 (Qns are also available in APAS)		
Practice Questions	Learn: using APAS system Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)		
Assignment	Learn: (1) Assignment Submission and Grading process; (2) Review Request Form (Procedure)		**Assignment paper – Available in APAS**			**Assignment due**	

Review Lecture – Week 8

- Course Introduction
- **Week 8 – Learning Materials**
  - Lectures
  - Lab and Tutorial
  - Code::Blocks IDE
  - LAMS MCQ Questions
  - Coding Practice Questions
  - APAS
- Reviews on Basic C Programming and Control Flow
- Examples

Lecture Video – Course Introduction

- **Watch Lecture Video** - Course Introduction on C Programming

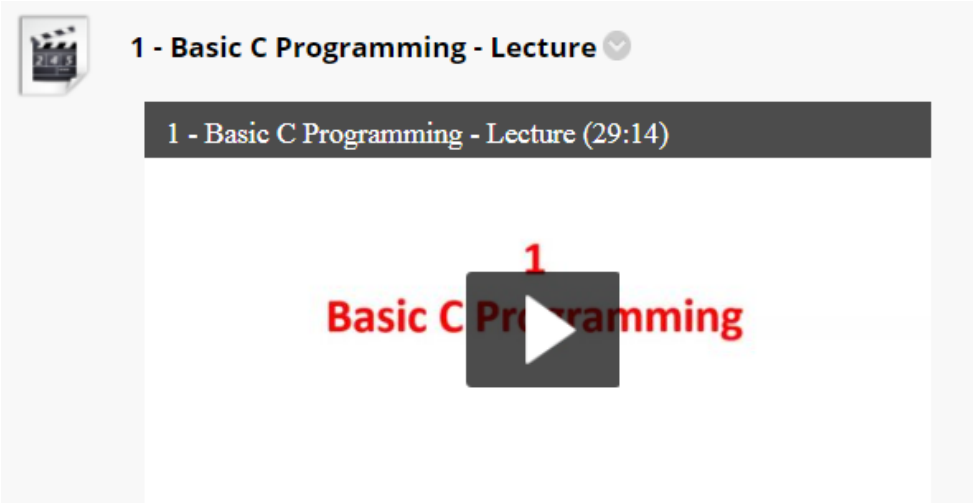
  (NTULearn: C Programming > E-Learning Lectures > Course Introduction)

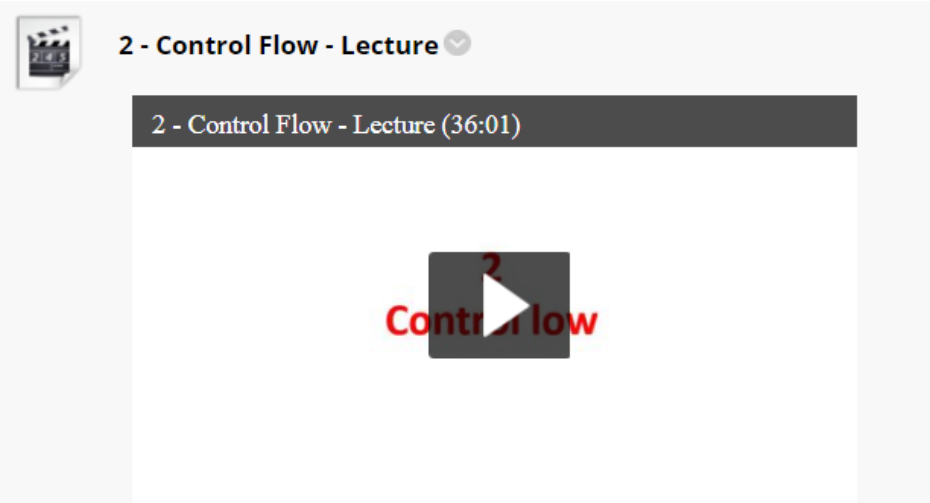
  - Topics for C Programming
  - Course Materials
  - Learning Schedules
  - Course Assessment
  - Integrated Development Environment (IDE) – Code::Blocks
  - Textbook for C Programming

Lecture Video – Basic C Programming and Control Flow

- **Watch Lecture Video - Basic C Programming and Control Flow**

(NTULearn: C Programming > E-Learning Lectures > Week 8)





Week 8 - Lab

- **Lab 1 – Basic C Programming & Control Flow**

(NTULearn: C Programming > Lab-Tutorials > Lab-Tutorial 1)

Lab 1 – Basic C Programming and Control Flow

Lab session – One hour is allocated for this lab session. There are 4 questions. The first two questions are lab questions. The last two questions are practice questions for you to try if you have extra time in the lab.

Note: You do not need to submit your code for this lab.

Lab Questions

1. Write a C program that prints the ID and grade of each student in a class. The input contains the student IDs and their marks. The range of the marks is from 0 to 100. The relationships of the marks and grades are given below:

Grade	Mark
A	100-75
B	74-65

Suggested solutions: Available in the same folder. You may refer to the suggested code if you have difficulty in attempting the lab questions.

Lab Coding Questions in APAS:

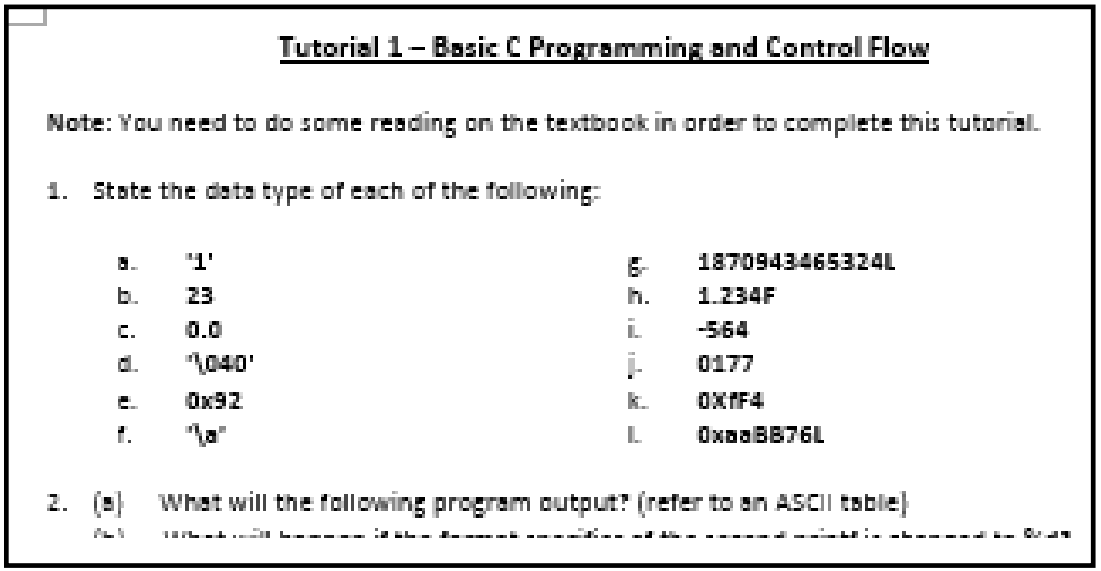
1. computeGrade
2. printAverage
3. printPattern
4. computeSeries

Available at: APAS > Exercise (choose Topic): You may test your code with sample test cases in APAS.

Week 8 - Tutorial

- **Tutorial 1 – Basic C Programming & Control Flow**

  (NTULearn: C Programming > Lab-Tutorials > Lab-Tutorial 1)



Tutorial 1 – Basic C Programming and Control Flow

Note: You need to do some reading on the textbook in order to complete this tutorial.

1. State the data type of each of the following:

a.	'1'	g.	1870943465324L
b.	23	h.	1.234F
c.	0.0	i.	-564
d.	'\040'	j.	0177
e.	0x92	k.	0XfF4
f.	'\a'	l.	0xaaBB76L

2. (a) What will the following program output? (refer to an ASCII table)

Suggested solutions: Will be available at the end of each week in the same folder.

Code::Blocks IDE

- **Read the document "Using Code::Blocks"** - To understand how to develop C programs under the Code::Blocks IDE (Integrated Development Environment).

  (NTULearn: C Programming > Lab-Tutorials)

Using Code::Blocks

Introduction to Code::blocks for lab sessions

Code::Blocks version in lab: 16.01

Code::Blocks current version: 17.12 (Requires to do some setting for debugger)

Basic creation of new project for coding

1. Launch Code::Blocks

LAMS MCQ Questions – Basic C Programming and Control Flow

- **LAMS MCQ Questions – Basic C Programming**

  (NTULearn: C Programming > LAMS MCQ Questions > Basic C Programming/Control Flow)

LAMS **LAMS MCQ Practice Questions - Basic C Programming**

Watch the lecture video and read the lecture notes before attempting the M
MCQ practice questions are for exercise only.

LAMS **LAMS MCQ Practice Questions - Control Flow**

Watch the lecture video and read the lecture notes before attempting
MCQ practice questions are for exercise only.

Basic C Programming

Q1

What will be the output of the program?

```
#include <stdio.h>
int main()
{
    printf("%f\n",2.5+1*7%2/4);
    return 0;
}
A. 2.500000
B. 2.750000
C. 3.375000
D. 3.000000
```

Control Flow

Q1

What will be the output of the program?

```
#include <stdio.h>
int main()
{
    int k,a=1,b=2;
    k=(a++==b)?2:3;
    printf("%d\n",k);
    return 0;
}
A. 0
B. 1
C. 2
D. 3
```

- Answers and explanations for each question are available in the same folder.

APAS - Coding Practice Questions

- **Coding Practice Questions – Basic C and Control Flow**

  (also available at APAS: Quiz > Basic C Programming and Control Flow)

1. temperature
2. powerLoss
3. cylinder
4. speed
5. distance
6. linearSystem
7. countChars
8. classifyChar
9. computeNetPay
10. computeSalaryGrade
11. computeCarPrice
12. printMultiTable
13. computeTotal
14. printPattern2
15. printPattern3

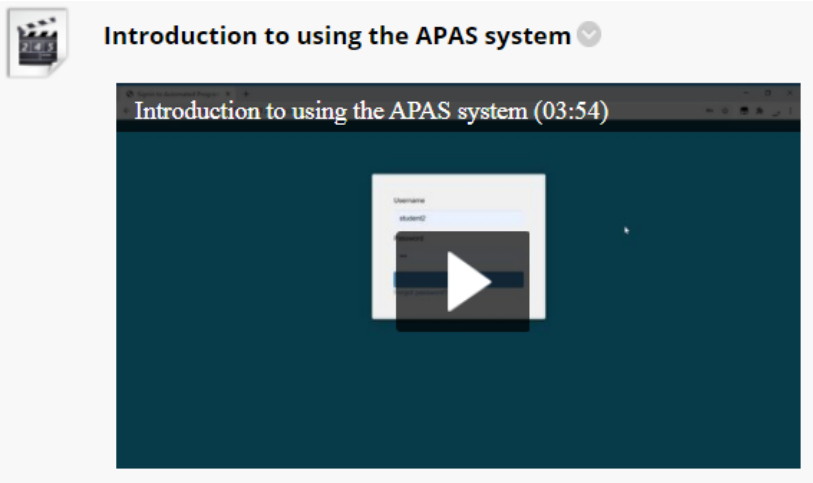
Suggested solution can be found at (VPN is needed when accessing from outside NTU):

http://172.21.147.174/ > NTUQA

APAS (Assignment Submission and Grading System)

- **Watch Video** – Introduction to using the APAS system

  (NTULearn: C Programming > Using APAS)

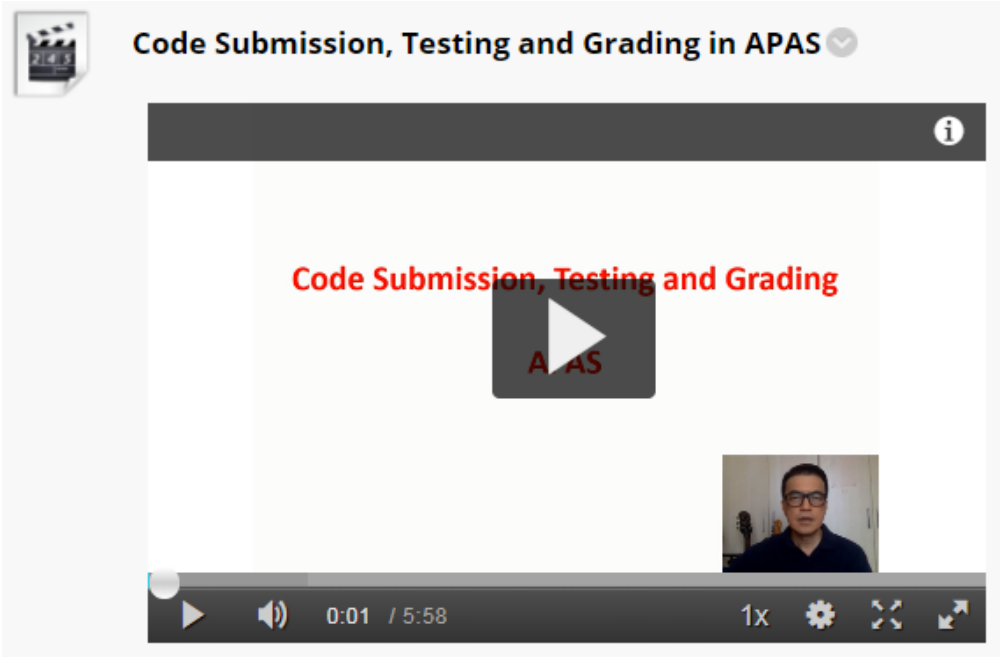


- **Read (doc)** – Introduction to using the APAS system

  (NTULearn: C Programming > Using APAS)

APAS – Code Submission, Testing and Grading

- **Watch Video – Code Submission, Testing and Grading in APAS (NTULearn: C Programming > Course Introduction)**



- Also refer to the slides on code submission and grading for APAS in the **Appendix**.

Review Lecture – Week 8

- Course Introduction
- Week 8 – Learning Materials
  - Lectures
  - Lab and Tutorial
  - Code::Blocks IDE
  - LAMS MCQ Questions
  - Coding Practice Questions
  - APAS
- **Reviews on Basic C Programming and Control Flow**
- Examples

Python vs C: Programming Languages

	Python	C
Definition	Python programs are saved by .py extension.	C programs are saved with .c extension.
	An **object-oriented** programming model is basically followed by Python.	An **imperative** (or procedural) programming model is basically followed by C.
Type	Python is dynamically typed.	C is statically typed.
Compilation	Python is an **interpreted** language.	C is a **compiled** language.
	Python is firstly compiled to a byte-code and then it is interpreted by a large C program.	C is compiled directly to machine code which is executed directly by the CPU.

Python vs C: Advantages of Python

	Python	C
Error Debugging	In Python, testing and **debugging** are directly not harder than in C.	In C language testing and debugging is harder.
Complexity	It is **easy** to learn, write and read Python programs than C.	Syntax of C is harder than python because of which programmers prefer to use python instead of C.
Built-in Functions	It is easy to implement **data structures** in Python with built-in insert, append functions.	Implementation of data structures requires its functions to be explicitly implemented.
Data Structures	Python has some **complex data structures** such as List, Dictionary, etc.	C does not have complex data structures.
Memory-Management	Python uses an **automatic garbage collector** for memory management.	In C, the Programmer has to do memory management on their own.

Python vs C: Advantages of C

	Python	C
Speed	Python programming language is slow.	C language is **fast**.
Pointers	No pointers functionality is available in Python.	**Pointers** are available in C language.
Applications	Python is a General-Purpose programming language.	C is generally used for **hardware related** applications.

Python vs C: Basic Syntax

	Python	C
Case-sensitivity	Python is **case-sensitive**. Y is not the same as y.	C is also **case-sensitive**. Y is not the same as y.
Where does execution start?	Executes the code in order from beginning to end of the file. main() is optional, and has no special meaning in Python.	Executes the body of **main()** and anything that main() calls. That means that C starts in the file in the project containing main(), but main() can call functions from other files.
Comments	From **#** to the end of the line	Begin with **/*** and end with ***/**, may span multiple lines. Many C compilers allow single-line comments that begin with **//**
Semicolons at the end of simple statements	Allowed, but only needed between two statements on the same line.	Required, as in x = f(a, b); It is mandatory to mark the end of every statement with a semicolon in C.
Indentation and line breaks	essential to the syntax of the language	Ignored by the compiler. It's still a good idea to have good, consistent indentation.

Python vs C: Variables and Types

	Python	C
Using code form other files	import module or from module import *	#include <stdio.h> or #include "myHeader.h"
Variable types	Variables are untyped; a given variable (label) can be stuck on values of different types at different times during the execution of a program.	The type of a variable must be declared when it is created, and only values of that type can be assigned to it. In C, we may not write x = 5.6; unless x has been previously declared: double x; ... x = 5.6; /* legal */ x = "String value"; /* This is illegal in C; types don't match*/
Variable declaration with assignment.	Python has no declarations	int n = 5; All variable definitions inside a function definition must occur at the beginning of a block of code, before any other statements.

Python vs C: Operators and I/O

	Python	C
Simultaneous assignment	x, y = 3, x-y	Not allowed in C
chained assignments	x = y = 0	x = y = 0;
Assignment with operators	x += 2	x += 2;
Increment operator	Not available in Python.	x++; ++x;
Exponentiation operator	x**3	No exponentiation operator in C: instead, use pow(x,3) defined in <math.h>

Reading numbers as input	m,n = input("Enter 2 numbers: ").split()
	print("sum = ", int(m)+int(n))
		int m, n;
		printf("Enter 2 numbers: ");
		scanf("%d %d", &m, &n);
		printf("sum = %d\n", m+n);
		
		If you don't use the & (address-of) operator, for each variable, it will not work.

Python vs C: Boolean Conditions

	Python	C
Boolean values	False True	**0 1** Actually, in C, any non-zero value is considered to be a true value, and zero false. There is no special Boolean type in C.
Relational operators	> < == != >= <=	> < == != >= <=
Logical operators	and or not	&& || ! The meanings are essentially the same, but the notation is different. We say "essentially" because the C operators produce integer values instead of special True and False values.

Python vs C: if Statements

	Python	C
basic if syntax	if <condition>: <one or more indented statements>	if (<condition>) <statement> Parentheses required. <statement> may be a single statement or a block of statements surrounded by { ... }.

if with else	if <condition>:
	  <one or more indented statements>
	else:
	  <one or more indented statements>
		if (<condition>)
		  <statement>
		else
		  <statement>

if with multiple cases	if <condition>:
	  <one or more indented statements>
	elif <condition>:
	  <one or more indented statements>
	else:
	  <one or more indented statements>
		if (<condition>)
		  <statement>
		else if (<condition>)
		  <statement>
		else
		  <statement>
		
		C has no special syntax that is similar to Python's **elif.**

Python vs C: switch Statement

	Python	C

switch statement
 	
Test a simple expression (for example, an integer), and choose which block of code to execute based on its value. If none of the specific cases match, execute the default code. If no default code and no cases match, do nothing.
	Python has nothing that is like C's **switch** statement.	**switch** (<expression>) {
		case <constant1> :
		<statement1>
		break;
		case <constant2> :
		<statement2>
		break;
		...
		default:
		<statement>
		}
		
		If there are multiple statements for a given case, no curly braces are needed.
		
		Don't forget the break; at the end of each case (except the last)

Python vs C: for Loop

	Python	C

for loop standard form	sum = 0
	for i in range(5):
	  sum += i
	  print(i, sum)
		int i, sum=0;
		for (i=0; i<5; i++) {
		  sum += i;
		  printf("%d %d\n", i, sum);
		}

More details of for loop syntax	for <variable> in <list:>: <one or more indented statements>	**for** (<initialize>; <test>; <update>) <statement> **initialize:** usually gives initial value to loop variable(s) **test:** If the value of the test is true, the loop continues. **update:** After each execution of the loop body, this code is executed, and the test is run again. <statement> may be a single statement or a block of statements surrounded by { ... }
A block of statements to be executed sequentially	indicated by indentation level	surrounded by { ... } In C, a block can be used anywhere a single statement can be used.

Python vs C: while Loop

	Python	C
do-nothing statement	pass	**;** In C, a single semicolon with nothing in front of it, indicates an empty statement. Just like Python's **pass**, used where the syntax requires a statement, but there is nothing to do there.
while loop syntax	while <condition>: <one or more indented statements>	**while** (<condition>) <statement> <statement> may be a single statement or a block of statements surrounded by { ... }.
break statement	break	**break;** Immediately end the execution of the current loop.
continue statement	continue	**continue;** Immediately end the execution of this time through the current loop, then continue the loop's next execution.

Review Lecture – Week 8

- Course Introduction
- Week 8 – Learning Materials
  - Lectures
  - Lab and Tutorial
  - Code::Blocks IDE
  - LAMS MCQ Questions
  - Coding Practice Questions
  - APAS
- Reviews on Basic C Programming and Control Flow
- **Examples**

Example 1 – boysGirls

Write a program that asks the user for the number of boys and that of girls in a class. The program should calculate and display the percentage of boys and girls in the class. A sample run is as follows:

Program Input and Output:

Enter the number of boys: *65*
Enter the number of girls: *77*
Boys: 45.77%
Girls: 54.23%

Example 1 – Suggested Code

Python Code:

```python
boy = 0
girl = 0
total = 0
percent_boy = 0.0
percent_girl = 0.0
# Get the number of boys.
boy = int(input("Enter the number of boys: "))
# Get the number of girls.
girl = int(input("Enter the number of girls: "))
# Calculate the total number of students.
total = boy + girl
# Calculate the percentage of boys.
percent_boy = boy / total
# Calculate the percentage of girls.
percent_girl = girl / total
# Print the percentage of boys.
print("Boys:", "{:.2%}".format(percent_boy))
# Print the percentage of girls.
print("Girls:", "{:.2%}".format(percent_girl))
```

C Code:

```c
#include <stdio.h>
int main()
{
  int  boy, girl, total;
  double percent_boy=0.0, percent_girl=0.0;

  printf("Enter the number of boys: ");
  scanf("%d", &boy);
  printf("Enter the number of girls: ");
  scanf("%d", &girl);

  total = boy + girl;
  percent_boy = ((float)boy/total)*100;
  percent_girl = ((float)girl/total)*100;

  printf("Boys: %.2f\n",percent_boy);
  printf("Girlss: %.2f\n",percent_girl);
  return 0;
}
```

Example 2 – FizzBuzz

Write a simple Python program to implement the FizzBuzz problem.

The Problem statement:

- Write a code that prints each number from 1 to 20 on a new line.
- Print “Fizz” if the number is the multiple of 3.
- Print “Buzz” if the number is multiple of 5
- For number which is multiple of both 3 and 5 print “FizzBuzz”

Program Output:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
```

Example 2 – Suggested Code

Python Code:

```python
for num in range(1, 21):
  if num % 15 ==0:
    print("FizzBuzz")
  elif num %3 ==0:
    print("Fizz")
  elif num %5==0:
    print("Buzz")
  else:
    print(num)
```

C Code:

```c
#include <stdio.h>
int main()
{
  int num;

  for (num=1; num<=20; num++)
    if (num % 15 == 0)
      printf("FizzBuzz\n");
    else if (num % 3 == 0)
      printf("Fizz\n");
    else if (num % 5 == 0)
      printf("Buzz\n");
    else
      printf("%d\n", num);

  return 0;
}
```

Example 3 – printOutput

Python Code:

```python
value = 6
if value % 2 == 0:
  print("first", value)
elif value % 3 == 0:
  print("second", value)

while value <= 9:
  value = value + 1
  if value == 8:
    continue
  else:
    pass
  print ("third", value)
else:
  print ("fourth", value)

print("fifth", value)
```

C Code:

```c
#include <stdio.h>
int main()  {
  int value;
  value = 6;
  if (value % 2 == 0)
    printf("first %d\n", value);
  else if (value % 3 == 0)
    printf("second %d\n", value);
  while (value <= 9) {
    value = value + 1;
    if (value == 8)
      continue;
    else
      ;
    printf("third %d\n", value);
  }
  printf("fourth %d\n", value);
  printf("fifth %d\n", value);
  return 0;
}
```

Program Output:

```
first 6
third 7
third 9
third 10
fourth 10
fifth 10
```

Program Debugging

Program Debugging Techniques:

- **Program Tracing** – add **print()/printf()** statements to the program and prints the internal states of the variables in order to trace the program.
- **Program Debugger** – **Code::Blocks** has program debugger, and you may try to use it for debugging the program. You will learn more about program debugger in the course on Data Structures.

Example 4 – printPattern

Write a Python program that reads an integer from the user, which is the width of the pattern below, and then prints out the pattern.

Program Input and Output:

```
Please enter pattern width: 5

*
**
***
****
*****
****
***
**
*
```

Example 4 – Suggested Code

Python Code:

```python
width = int(input("Please enter pattern width: "))

for i in range(width+1):
    #print(i, sep=' ')
    for j in range(i):
        print("*",end="")
    print()
for i in range(width-1,0, -1):
    #print(i, sep=' ')
    for j in range(i):
        print("*",end="")
    print()
```

C Code:

```c
#include <stdio.h>
int main()
{
  int i, j, width;

  printf("Please enter pattern width: ");
  scanf("%d", &width);

  for (i = 0; i < width+1; i++) {
    for (j = 0; j<i; j++)
      printf("*");
    printf("\n");
  }
  for (i = width-1; i>0; i--) {
    for (j = 0; j<i; j++)
      printf("*");
    printf("\n");
  }
  return 0;
}
```

Example 5 – studentGrade

Write a C program that prints the ID and grade of each student in a class. The input contains the student IDs and their marks. The range of the marks is from 0 to 100. Use the sentinel value –1 for student ID to indicate the end of user input. The relationships of the marks and grades are given below:

mark	Grade
80 <= mark	A
70 <= mark < 80	B
60 <= mark < 70	C
50 <= mark < 60	D
40 <= mark < 50	E
mark < 40	F

Program Input and Output

Enter StudentID: *11*
Enter Mark: *56*
Grade = D
Enter StudentID: *21*
Enter Mark: *89*
Grade = A
Enter StudentID: *31*
Enter Mark: *34*
Grade = F
Enter StudentID: *-1*
Program terminating ...

Example 5 – Suggested Code

Python Code:

```
studentID = int(input("Enter StudentID: "))
while studentID != -1:
  mark = int(input("Enter Mark: "))
  if mark >= 80:
    grade = 'A'
  elif mark >= 70:
    grade = 'B'
  elif mark >= 60:
    grade = 'C'
  elif mark >= 50:
    grade = 'D'
  elif mark >= 40:
    grade = 'E'
  else:
    grade = 'F'
  print("Grade = ", grade)
  studentID = int(input("Enter StudentID: "))
```

C Code:

```
#include <stdio.h>
int main() {
  int studentNumber = 0, mark;
  char grade;
  printf("Enter StudentID: ");
  scanf("%d", &studentNumber);
  while (studentNumber != -1)  {
    printf("Enter Mark: ");
    scanf("%d", &mark);
    if (mark >= 80)      grade = 'A';
    else if (mark >= 70)       grade = 'B';
    else if (mark >= 60)       grade = 'C';
    else if (mark >= 50)       grade = 'D';
    else if (mark >= 40)       grade = 'E';
    else grade = 'F';
    printf("Grade = %c\n",grade);
    printf("Enter StudentID: ");
    scanf("%d", &studentNumber);
  }
  return 0;
}
```

Example 5 – Suggested Code with switch

```
...
while (studentNumber != -1)  {
    printf("Enter Mark: ");
    scanf("%d", &mark);
    switch ( mark/ 10 )  {
      case 10: case 9: case 8:
        grade = 'A';
        break;
      case 7:
        grade = 'B';
        break;
      case 6:
        grade = 'C';
        break;
      case 5:
        grade = 'D';
        break;
      case 4:
        grade = 'E';
        break;
      default:  grade = 'F';
    }
```

C code using switch

Using integer division (gives integer result in C):

85/10 -> 8 64/10 -> 6

87/10 -> 8 68/10 -> 6

74/10 -> 7 34/10 -> 3

…

Division by 10 forms 11 categories from marks:

0, 1, 2, 3, … 10

Appendix

APAS
Programming Submission and Grading System

APAS - Login

- **APAS: http://172.21.146.80/**
- An online Automatic Programming Assessment System (**APAS**) will be used for this course for Exercise, Quiz (Practice), Assignment and Test.
- Please use Chrome or Firefox for accessing APAS.
- Log in with your network account in uppercase:
  - For example, your email is sd0001@e.ntu.edu.sg
  - Your username for APAS will be: **SD0001** (UPPERCASE)
  - Password: **SC1003** (Change your password after 1st login)
- If you want to access APAS from outside of NTU, you will need vpn (e.g. use Pulse Secure) for accessing it.
- After logging into the system, you may select the Exercise, Quiz, Assignment or Test for program editing, submission and grading.

APAS - Program Testing

After creating the program, you may test the program with the following three options:

1. "**Try Compilation**" – It compiles your program and detect compilation errors.
2. "**Test with Sample Inputs**" - The question requirements will provide you with sample input/output data 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.
3. "**Run Input**" - 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.

APAS - Automated Program Grading

- The answer code can be submitted to APAS and marked automatically. The grading is achieved through the checking of input/output data from **test cases**.
- The grading is carried out based on program input/output data of test cases with the "**exact string matching**" technique.
- Please note that the **sample test cases** (used in **pretest**) are only used for **testing only**. No scores will be assigned to the correctness of running sample test cases.
- We use **"hidden" test cases** for the grading of your program as this is to ensure that your program code will not be hard-coded to achieve the desired output results.
- Note that the final score is computed according to the correctness of the programs based on "**hidden**" test cases, **NOT sample test cases**.

APAS - Notes on Coding

1. Do not change the **main()** function in the program template.
2. Each assignment question will provide you with the problem specification, program template and sample input/output printout sessions. These are the problem requirements. **You should follow exactly the problem requirements to build your programming code for the question**.
3. Therefore, it is extremely important to pay special attention to the **program input and output data format** (including letter cases) when you write your programs (a simple mistake on letter case will cause your program to be marked as incorrect in APAS) as we use **exact string matching** technique for checking the correctness of the program.
4. As such, you have to follow exactly the printout format shown in the sample input/output sessions when writing your programs.

APAS - Notes on Coding

- Compilers in APAS and Code::Blocks are different, so note that:
  - No inline declaration in C (i.e. declare all variables at the beginning of the function code).
  - Initialize local variables before use if the initial values are needed for computation.
- **"TIMEOUT"** error message – it occurs when the program waits for user input and no input has occurred. Therefore, you need to check whether you have provided all the input data when running the program.
- Also note that there is **NO need to do user input error checking** in your program unless it is explicitly stated in the question requirement.

Thank you !!!