

### 1.1-1.2 Software Life Cycle

The purpose of good **design** is to change a **vague problem** into a very **specific solution**.

Every computer **system** has 4 important components:

[ Input ] -----> [ Storage ] <=> [ Processing ] -----> [Output]

During the analysis and design phases, the **analyst/designer** must sort out the things which belong in each section. He or she needs to produce answers to the following questions:

#### Input

- What data will be input into the system – get **sample** data and describe **types, records, fields**
- **How much** data will there be? **How often** will input occur?
- Where will it come from – out the user's head, from a sheet of paper, from another computer?
- What is the best way to **capture it** – what device – Mouse? Keyboard? Scanner? Other?

#### Storage

- What data will be **stored** permanently? How long must it be stored? What can be thrown away?
- **Where** should the data be stored – Hard-disk, tape, floppy, server, web-site?
- **How much** data will be stored total? What total **storage capacity** is needed?
- What **data files** will be needed? What will the **structure** of each file be?

#### Output

- What data will be output from the system?
- What **output devices** are needed – monitor? printer? sound? other?
- What **reports** must be produced? How should they be **formatted**? Produce **sample reports**.
- Does any output get **reused** by another system or by this system? -> computer readable output.

#### Processing

- What **processes** must occur to change the **available input** into the **required output**? Fixing bad data? Calculations? Sorting? Filtering? Reorganizing? Reformatting?
- Which processes should be **automated**? Which processes must be performed by the **user**?
- What **processing devices** are needed – faster CPU? Special math capabilities? Multimedia capabilities (sound, graphics)?

#### QUESTIONS: Airplane Reservations

- Describe three significantly different **types of data** in this system.
- Describe three significantly different **users** of this system.
- For one of the **types of data**, estimate the **total quantity of input** required per week.
- Describe two uses of a **computer terminal** by a travel agent in this system.
- Tickets can be printed at the travel agency. What **output device** is required? Does it have any special requirements other than normal functions?
- Describe a **processing feature** in this system which is **automated** and could probably not be done **manually**.
- If a customer is to be **reminded automatically** of an upcoming flight, describe the **processing** required for this reminder, including **accessing stored data** and **producing output**.
- A **flight attendant** needs to **input** data into this system – what data? Under what circumstances?
- Explain what a **data-flow-diagram** is, and draw one for the process of **checking-in** at the airport.

**STAGES** in system development:

The **investigation/analysis** stage should produce a **feasibility study** which involves:

- **ANALYSTS (and USERS)**
- User's goals and requests
- Stories from users, describing problems and needs
- Sample data
- Possible hardware choices (perhaps several alternatives)
- Estimated costs and/or financial limitations

The **design** phase makes a **connection** between the **requirements (user's needs)** and the **implementation (programmer's solution)**. The design phase produces a **requirements document** including:

- **DESIGNERS (and USERS)**
- **Features** which the solution must have
- **Tasks** which the user should be able to perform using the system
- **Hardware** requirements or limitations (e.g. small portable computer are required in a warehouse)
- Any other requirements and/or limitations
- making sure the requirements fit within any **limitations** stated in the **feasibility study**

The **implementation** stage involves:

- **SOFTWARE and HARDWARE ENGINEERS (no USERS, but maybe "beta" testers)**
- breaking the problem into **modules** which can be constructed, tested, and debugged easily
- choosing, buying, or building hardware
- choosing a programming language
- writing a program which implements the modules
- debugging the **modules** and the finished **system**.
- producing **documentation** of any limitations or requirements of the solution
- **ALWAYS** making sure the resulting system satisfies the **requirements specification**

**Installation and operation** involves:

- **TECHNICIANS (preferably not USERS)**
- purchasing necessary hardware
- installing and configuring the software
- training the users
- **ALWAYS** following any **instructions** presented in the **documentation** from above

**Maintenance and review** involve:

- **TECHNICIANS, USERS, and SUPERVISORS**
- talking to and helping **users**
- fixing or replacing broken components - hardware or software (sometimes defective users)
- doing backups, and restoring backups after data gets lost
- monitoring the growth of data and avoiding system problems like full servers
- reviewing the system by observing and recording its performance, reliability, and productivity
- producing **proposals** for system improvements and modifications

People in one stage may **cycle back** to those in the previous stage and ask for **changes** if there are problems.

-- **Explain two reasons** why it is desirable to have **different people** involved in the different stages.

-- In which stage does **module-level testing** occur?

-- In which two stages would a **questionnaire** be useful?

### 1.3 Software Design

- # Explain what **OMR** is, and give an example of where it is commonly used.
- # Explain what a **GUI** is, and describe an **alternative** which is commonly used.
- # Describe 3 different **variable types** which could be used to store a **date** in a program – **not** java Date type.
- # State a whole number which **cannot** be stored in an integer variable.
- # State a **floating point** number which **cannot** be stored in a **real** variable.
- # State an **ASCII code** which does **not** represent a character.
- # Outline an example of a sensible use of an **array** for storing data, and explain how an **error** might occur as a result of this application.
- # Outline an example of using **parallel arrays** to store data.
- # Outline a situation where a **sequential file** should **not** be copied into an array.
- # Describe an application where a **2-dimensional array** should be used for data storage.

A common **algorithm** is the **bubble-sort**.

- # Explain how it is possible for a bubble sort to stop after the first pass through the data.
- # Explain how a single bubble-sort procedure could be used to sort 3 different arrays.
- # If a bubble sort successfully sorts 1000 items in 10 seconds, how long should it take to sort 8000 items?
- # Which is generally faster – a **bubble sort** or a **selection sort**?
- # If the items in a list are already sorted, which will finish faster – **selection** or **bubble sort**?

Another common algorithm is a **sequential search**.

- # Explain how it is possible for a sequential search to stop before it has examined the entire list.
- # Which should run faster – a sequential search in an array, or a sequential search in a file?
- # Describe an **alternative** search algorithm.
- # If a sequential search requires 10 seconds to search through a text file containing 1000 names, approximately how long would it take to search through a text file containing 8000 names?

### 1.4 Constructs

- # What does **scope** of a variable refer to?
- # Which has larger scope – local variables or global variables?
- # Does a **parameter** behave more like a local variable or a global variable?
- # Give an example of a **user-defined** data type.
- # What is a **relational** expression? Give an example.
- # What is a **boolean** expression? Give an example.
- # What is the correct **operator precedence** of Boolean operators, comparison operators and arithmetic operators in PURE?
- # A **record** can contain many different types of data in different \_\_\_\_\_.
- # Give an example of a **selection construct**.
- # Give 3 examples of **looping constructs** available in PURE.
- # Explain what an **iteration construct** is.
- # What are **sentinel values** used for?
- # Give an example of a **flag**.
- # Explain the difference between passing parameters **by-reference** or **by-value**.

## Long Question

LIST is a **string array** containing names students. The end of the array is marked by the entry "XXX".

(a) Write a **function** COUNT which reads the LIST until the "XXX" is found, and returns the NUMBER of names in the list. It should start in location [1], and should **not** include the "XXX" in the count. So if location [1] contains "XXX", the result should be 0. Your function **must** use a **While..** loop to do this task.

(b) Write a **procedure** REVERSE which uses the COUNT function (above), and prints the all the names in the list in **reverse order** (starting before "XXX", and ending at location [1]). Your procedure **must** use a **For..** loop to do this task.

## 1.4 Errors and Testing

- # Write a function which adds up the series  $1 + 2 + 3 + 4 + \dots + N$ , where N is a parameter.
- # If your procedure produces an incorrect answer, is that a run-time error, a syntax-error, or a logic-error?
- # If one single **character** (letter) were accidentally erased from your program, what type of error would result?
- # If the value of the parameter N is set to equal 0 (zero), would this cause a run-time error?
- # Explain why a **compiler** can detect syntax errors, but cannot prevent run-time errors.
- # Describe how you would **test** your program, including 3 **test-cases** which you would use.
- # State a value for the parameter N which might cause a **run-time** error.  
Explain how this error could be **handled** (detected and/or prevented).
- # Describe three features or tools in Visual Basic which make **debugging** a program easier.
- # Describe what the term **Dry-run** means, and how this can be accomplished without using a computer.
- # If an infinite loop runs "forever", what kind of an error is this?

## Longer Question

Consider the following algorithm:

```
public static void main String[] args)
{
    output ROUNDED(19.9559);
}

public int ROUNDING(int NUM)
{
    return (int)Math.floor(NUM + 0.5);
}
```

- # The algorithm contains a **syntax error**. State the cause of the error and **fix** it.
- # Explain why the algorithm produces a **type-mismatch error**. Then **fix** it.
- # State the output of the algorithm after your corrections.
- # The user **expected** the algorithm to round the number to 2 decimal places, producing 19.96. The output is incorrect. Which type of error is this?

## 1.5 Documentation

- # Explain 3 differences between **internal** documentation (read by programmers) and **user documentation**.
- # User documentation is usually **not** written by programmers. Who should (would or could) write this documentation and why would they do a better job than a programmer?

## 2.1 Language Translators

The NEW programming language (Network Enabled Writing) is going to change the way programmers work. The designers noticed that there are lots of good programmers living in India and other low-cost economies, who can successfully write software for use in more expensive industrialized countries (e.g. Germany). The programmers in India will work for about half the salary that programmers in Germany demand.

Software houses can use E-mail to send source code (programs), requirements specifications, sample data, and documentation back and forth. This works reasonably well for small projects, where only a few programmers are involved. But on big projects, the programmers really need to work together very directly. They write procedures and want to test them immediately in the larger project. Until now, this required the programmers to be at one location – e.g. in Germany - as the application must be developed in the environment where it will be used.

The NEW language will enable remote (distant) programmers to write code and test it in the larger project immediately, but this is difficult to manage. For example, if the program needs to **open** a data-file, it will need to open the file on a server in Germany, using the Internet to make the connection. An even bigger problem is **compiling**. The compiler will run on the client in India, but must access other modules in the server in Germany.

The NEW Team (NEWT) is designing the language to be very similar to Java, but with some **extensions**. One specific extension is the **REMOTE** construct. By adding **REMOTE** to a command, the compiler understands that it must access something on the German server. All the **handshaking** issues are preformed automatically to make the connection, and the compiler will **wait** for notification before proceeding. The **REMOTE** command is used as illustrated below:

Local (normal) Version	NEWS (Remote) Version
<pre>import ECOMMERCE;  void UPDATE(double AMOUNT, int ID) {     boolean OKAY = true;     double MONEY;     MONEY = GETBALANCE (ID);     if (MONEY &gt;= AMOUNT)     {    POST (AMOUNT, ID);    }     else     {    LOCK (ID);    } }</pre>	<pre>remote import ECOMMERCE;  void UPDATE(double AMOUNT,int ID) {     boolean OKAY = true;     double MONEY;     MONEY = GETBALANCE (remote ID);     if (MONEY &gt;= AMOUNT)     {    remote POST (AMOUNT, ID);    }     else     {    remote LOCK (ID);    } }</pre>

```
library ECOMMERCE contains
/*****
    public void POST (AMOUNT, ID)
        adds or subtracts money in an account for the customer ID

    public double GETBALANCE (ID) returns real
        opens the customer accounts file and finds out the amount of
        money in the customer's account

    procedure LOCK (ID)
        locks a customer account to prevent money from being spent -
        perhaps if the customer's balance is negative
*****/
```

Notice that the **remote** command only applies to single commands, not the entire program. For example, the variable **AMOUNT** is always stored and processed locally, whereas the command **LOCK(ID)** might be executed on the server instead. Performing **LOCK(ID)** locally would be pretty useless, as the server is not going to look on the Indian programmer's computer to find out whether an account is locked. Each time a **remote** command is executed, the local client must **make a connection** with the server in Germany. After that command, the connection is immediately broken.

NEW Team is having an argument about how and where to write the **REMOTE** command in a listing. It is in a strange place in this command: `MONEY <-- GETBALANCE (remote ID)`  
 In other commands it appears at the beginning.

Some of the Germans in the group are also not happy with the word "remote" – they would rather have an English word like **EXTERNAL** which has a similar word in German (**EXTERN**).

- # Is the **position** of the **remote** identifier a **syntax** issue, a **semantics** issue, or **neither**?
- # Does the choice of the word **remote** or **extern** represent a **syntax** issue, a **semantics** issue, or **neither**?
- # One of the NEWTs said he would rather use the symbol **\$** to represent a **remote** function, rather than typing a whole word. Is this a **syntax** issue, a **semantics** issue, **neither**, or **both**?
- # If the Indian programmer uses an **interpreter** rather than a **compiler** for NEW, explain why this choice could make things a lot slower, whereas this choice makes little difference if he is working **locally** with Visual Basic.
- # Is NEW intended for use in a LAN, a WAN, both, or neither?
- # Explain why it would not be possible to use HTML instead of using NEW.
- # A CASE tool can produce a **system flowchart** for the entire ECOMMERCE project. Is it better for the CASE tool to run locally on the client, or should it run on the server?

=====

## 2.2 Architecture

Match each device with an application.

Device	Application
Mouse	change a printed page back into a word-processing document
Keyboard	respond to movement or light signals
touch-screen	create bar-coded sticky labels for library books
OCR	print an ID number on a bank cheque
MICR	select from a drop-down menu on a PC
scanner	copy a photograph into the computer
LCD panel	transmit packets of data from one computer to another
voice recognition	draw accurate pictures with a stylus
sensors	convert analog sound to digital data and transmit it from various locations
graphics tablet	display data in a portable computer
printer	give commands in a command-line interface
plotter	capture pictures which can be displayed over the Internet
sound card	give commands to a computer when your hands are busy
digital camera	input music
cellular phone	draw accurate architectural diagrams, without jaggies
modem	start a program in a pocket-organizer

### 2.2.2 Chips

The ALU and CU are two parts of the \_\_\_\_\_. The ALU performs \_\_\_\_\_ and \_\_\_\_\_ operations. The \_\_\_\_\_ controls communications across the data bus.

The data bus contains 32 (or maybe 64) \_\_\_\_\_ circuits, each of which can carry one \_\_\_\_\_ of data.

The CPU uses the data bus to fetch and store data in the \_\_\_\_\_.

But to speed up operations, the data bus does not connect directly between the CPU and the RAM.

The CPU uses the \_\_\_\_\_ for temporary storage, and **it** is connected to the RAM.

The speed of the CPU might be 1 \_\_\_\_\_ Herz. This means \_\_\_\_\_ cycles per second.

If the data bus is 64 \_\_\_\_\_ wide, and runs at 266 \_\_\_\_\_,

then it can transfer data at a speed of \_\_\_\_\_ MegaBytes / Sec.

This is the same as \_\_\_\_\_ GigaBytes per second.

Earlier computers did not run at such high speeds. Running a CPU at a higher speed generates a lot of \_\_\_\_\_.

Cooling fans can only help a certain amount. Super computers use liquid nitrogen to cool the circuits and chips,

which can then run at very high frequencies without damage. In PCs, the solution is to make the circuits in the

chips \_\_\_\_\_, reducing the size from 0.25 microns to 0.13 microns, and then also reducing the

voltage from 3 V to 1.5 V. Using less energy produces less heat, so the chip can run at a faster frequency

without overheating. It is also possible to place more \_\_\_\_\_ on the chip in the same space.

A Pentium chip contains over 25 \_\_\_\_\_ transistors.

In RAM, a transistor represents one \_\_\_\_\_. 8 \_\_\_\_\_ makes 1 \_\_\_\_\_.

Modern PCs contain 128 \_\_\_\_\_ of RAM, or more. A hard-disk might store 40 \_\_\_\_\_ of

data. The hard-disk can store \_\_\_\_\_ times as much data as the RAM. When a program runs, it must

be copied into the \_\_\_\_\_. If several programs are running at once (\_\_\_\_\_),

the RAM could get full. Then the CPU must **swap** some of the memory out onto the \_\_\_\_\_,

to free up more memory for other programs. This process is called \_\_\_\_\_ memory, because the

computer seems to have more memory than the actual physical RAM.

**2.2 Storage**

T = Tera = 2 ^ \_\_\_\_\_ 1 Terabyte is equal to \_\_\_\_\_ MegaBytes.

G = Giga = 2^ \_\_\_\_\_ A 40 GigaByte hard disk can hold as much as \_\_\_\_\_ floppy diskettes.

M = Mega = 2^ \_\_\_\_\_ Using a 56 K modem, the fastest possible download of 1 MB is \_\_\_\_\_ sec.

K = Kilo = 2^ \_\_\_\_\_ A hard-disk sector containing 512 bytes is the same as \_\_\_\_\_ KB.

A 32 bit address bus can carry addresses between 0 and \_\_\_\_\_, so it can address a maximum of \_\_\_\_\_ RAM.

A \_\_\_\_\_ interface has only 1 data wire, and transmits 1 bit after the next.

If it runs at 240 KHz, it can transmit \_\_\_\_\_ KB per second.

A \_\_\_\_\_ interface uses 8 data wires, and thus can transmit an entire byte at one time.

- =====
- # Explain why a Kilometer is exactly **1000** meters, whereas a KiloByte is **1024** bytes.
  - # A diskdrive can access files using either sequential or direct-access. Explain the difference.
  - # If direct access is faster than sequential access, then why do would a backup system use sequential access to backup 10 GigaBytes of data?
  - # What technological improvements enable modern computers to contain 256 MB of RAM for roughly the same price as a computer 3 years ago which had 64 MB of RAM?
  - # A DVD disk is the same physical size as a CD-ROM. The CD-ROM stores 700 \_\_\_\_\_. The DVD stores 4.9 \_\_\_\_\_. One DVD can store the same amount as \_\_\_\_\_ CDs. How does the DVD manage to store so much more data?

- =====
- # Explain the difference between **sequential** access and **direct-access** in a data file.
  - # Which is usually faster – sequential or direct access?
  - # Why does **direct access** requires **fixed-size records**, while **sequential access** permits **variable size records**?
  - # Explain why the backup process for a **10 GB** hard-disk is likely to use **sequential access**.

**Long Question**

- A new **pocket PC** contains no disk drive. Instead it uses **flash memory**. This is a large amount of **RAM** (e.g. 64 MB) with a constant power supply, so the data never gets erased. A special design achieves this without using very much power, so batteries can last several weeks or months. But flash memory is expensive.
- # The operating system is stored in 16 MB ROM. Why not store the OS in RAM instead?
  - # Describe an appropriate backup device for this pocket-sized computer.



- # The LCD screen goes blank after 1 minute if the user doesn't do anything. Why is this important?
- # Even with a modem connection, the pocket-PC does not display web-sites sensibly. Why not?
- # Why is a voice-recognition system more useful in a pocket-PC than in a desktop PC?

### 2.3 Systems

- # Which of these applies to the Windows operating system?  
**Real-time    Multi-tasking    GUI    Multi-user**
- # Why would a different OS be used in a web-server (e.g. Unix) than in a PC (e.g. Windows)?
- # What is the name for a system where more than one program can run at the same time in one computer?
- # Why is a **main-frame** more likely to contain multiple processors than a PC ?
- # List 3 different things which can be shared in a LAN.
- # Describe two different types of computers involved in E-mail (with two fundamentally different functions).
- # What is the main difference between **on-line processing** and **batch processing**?
- # Describe a typical **batch process** performed by a bank computer.
- # Describe a typical **on-line process** performed by a bank computer.
- # Describe a typical **batch process** performed by a library computer.
- # Describe a typical **on-line process** performed by a library computer.
- # Explain why there are no important **real-time processes** for a library computer.
- # Explain why an **air-traffic-control** system must function in real-time.
- # Which is the most likely processing mode of a hospital intensive-care computer – **online, batch, or real time?**

### Master and Transaction Files

A **master file** is the main file with the current data in it. In a store, the names of the products and the prices and the number on the shelves (inventory) are in the master file. If the store receives a **delivery**, that might be a truck with lots of toilet paper, cola bottles, lettuce, etc. The delivery will come with a **list** of the items. The fact that these are delivered is a **transaction** – a change in the data. If they are lucky, the store receives the list of items (transactions) on a computer readable diskette or card or tape. Then they can **post** these transactions into the **master** file, which changes the inventory (stock levels):

<b>Master File</b>	+	<b>Transactions</b>	====>	<b>NEW Master File</b>
<b>Inventory</b>		<b>Delivery</b>		<b>New Stock Levels</b>

### 2.4 Data Representation

#### Analog/Digital

Which of the following involve DAC or ADC? Which do not?

- |   |  |
|---|--|
| <ul style="list-style-type: none"> <li>- Storing data on a disk drive</li> <li>- Playing a phonograph record on a stereo</li> <li>- Using a phone-card in a public telephone</li> <li>- Printing from a PC to a laser printer</li> <li>- Photography with a digital camera</li> </ul> | <ul style="list-style-type: none"> <li>- Playing CD music on a PC speaker</li> <li>- Measuring temperatures and storing them in a data file</li> <li>- Controlling an industrial robot</li> <li>- Audio sampling (what's that?)</li> <li>- Scanning and OCR</li> </ul> |
|---|--|

#### Binary

- # 24-bit color is called **true color**. How many different colors are there?
- # One byte can contain any positive number between \_\_\_\_\_ and \_\_\_\_\_.
- # How many bits are needed to store the number 1,000,000 ?
- # Write the following IP address as a 32-bit binary number: 179.123.10.99
- # If the byte 10101100 is transmitted, what **parity bit** should be transmitted with it (even parity)?
- # Explain what a **checksum** is. Is it most appropriate with a byte, a packet, a file, or a disk drive?

**Long Question A**

A clever computer user named **Nerdo Jones** heard about **compression** programs. He figured out that he could compress his e-mails to about 50% the original size. He figures this will get them to their destination faster. Nerdo wrote his own program to do the compression. Then he compressed a few E-mails and sent them to friends. Here is the algorithm:

```
class CompressA
{ int[] CODES = new int[1000];

  public static void main(String[] args)
  { COMPRESS("NERDOS"); }

  public void COMPRESS(String MESSAGE)
  {
    int A,B,C,D;
    char LETTER;
    C = 0;
    while (C < MESSAGE.length())
    {
      LETTER = MESSAGE.charAt(C);
      A = ((int)LETTER) % 16;
      LETTER = MESSAGE.charAt(C+1);
      B = ((int)LETTER) % 16;
      D = 16*A + B;
      CODES[(C+1)/2] = D ;
      C = C + 2;
    }
  }
}
```

Recall that the ASCII code of "A" is 65, and the ASCII code of "Z" is 90.

- Trace the algorithm and state the contents of the array CODE after COMPRESS("NERDOS") runs.
- Explain why this compression algorithm shrinks the size of the message by 50%.
- Outline two reasons why the recipients of Nerdo's coded messages will **not** be able to decompress them.
- State a value of MESSAGE which probably causes a run-time error, and explain why.

**Long Question B**

A text file contains many lines of text – an e-mail message. Nerdo has purchased a proper compression routine called **SQUASH**, which can compress a string. He wants to use the SQUASH function to compress the entire file. Assume the method OPEN exists and opens a text-file for reading and/or writing.

```
void COMPRESSFILE() throws IOException
{ String WORDS = "";
  String COMP = "";
  BufferedReader EMAIL = open("EMAIL.txt");
  BufferedReader CMAIL = open("CMAIL.txt");
  while (EMAIL.ready())
  { WORDS = EMAIL.readLine();
    COMP = SQUASH(WORDS);
    EMAIL.println(COMP);
  }
  EMAIL.close();
  CMAIL.close();
}
```

This seems to work fine, but Nerdo is unhappy that the resulting file CMAIL is not the same file as the original file. Rewrite COMPRESSFILE so that it reads the entire EMAIL file into an array, then SQUASHES the contents of the array, and then outputs the compressed strings back into the original EMAIL file.

## 4.1 Numbers

### Short questions

- =====
- Calculate the largest unsigned integer which can be stored in 24 bits.
  - Change the **two's complement** binary number 10101100 to decimal.
  - Change the **unsigned** binary number 10101100 to decimal.
  - Change the decimal number 999 to a 16-bit binary number.
  - Change the hexadecimal number AB90 to unsigned binary.

Add the unsigned binary number 10010011 + 00001111, give the answer in binary.  
 Explain the "shortcut" for multiplying a binary number by 8.

- Write the decimal 3.25 in binary.
- Explain why it is **not possible** to store the decimal value 1/10 correctly in binary.
- Determine the mantissa and exponent of 1234.56 in normalized scientific notation.
- Explain why the number 1.5 only requires 2 bits for storage.
- Explain why it would be a **bad idea** to store 1.5 in 2 bits.

## 4.3 Modulo Arithmetic (mod,div)

### Short Questions

=====

Calculate the value of  $(12345 \text{ div } 1000) \text{ mod } 100$ .


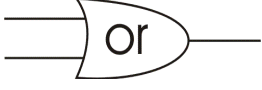
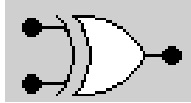
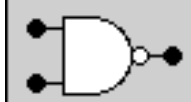
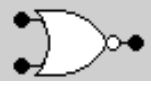
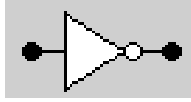
- Calculate each of the following:
- $12345 \% 10$  ,  $12345 / 10$
  - $1234 \% 10$  ,  $1234 / 10$
  - $123 \% 10$  ,  $123 / 10$

Show how to write a **loop** to execute the calculations above, and to **add up** the digits in an integer number.

Change your previous answer to use a **weighting** factor for each digit, performing a **check-sum** calculation:  
 $12345 \implies 1*3 + 2*5 + 3*7 + 4*9 + 5*11 \implies 125$   
 where the odd numbers 3,5,7,9,11 are the "weighting factors".

Explain why the use of **weighting factors** provides a more reliable error-checking method rather than simply adding up the digits.

**4.2 Boolean Logic**

Operator	Logic Circuit Gates	Boolean Algebra	Boolean Expression	Truth Table Values 0 = False, 1 = True
<b>AND</b>		$p \cdot q$	$p \text{ AND } q$	$0 \text{ AND } 0 = 0$ $0 \text{ AND } 1 = 0$ $1 \text{ AND } 1 = 1$
<b>OR</b>		$p + q$	$p \text{ OR } q$	$0 \text{ OR } 0 = 0$ $0 \text{ OR } 1 = 1$ $1 \text{ OR } 1 = 1$
<b>XOR</b>		$p \oplus q$	$p \text{ XOR } q$	$0 \text{ XOR } 0 = 0$ $0 \text{ XOR } 1 = 1$ $1 \text{ XOR } 1 = 0$
<b>NAND</b>		$\overline{p \cdot q}$	$\text{not}(p \text{ AND } q)$	$0 \text{ NAND } 0 = 1$ $0 \text{ NAND } 1 = 1$ $1 \text{ NAND } 1 = 0$
<b>NOR</b>		$\overline{p + q}$	$\text{not}(p \text{ OR } q)$	$0 \text{ NOR } 0 = 1$ $0 \text{ NOR } 1 = 0$ $1 \text{ NOR } 1 = 0$
<b>NOT</b>		$\overline{p}$	$\text{not } p$	$\text{NOT } 0 = 1$ $\text{NOT } 1 = 0$

**Short Questions**

State the correct **order of operations** (operator precedence) for the 6 operators above.

Write a **truth table** showing all the possible values of :  $(p \text{ AND } q) \text{ OR } (p \text{ XOR } q)$

Draw a circuit which is equivalent to :  $\text{not}(a \text{ AND } b) \text{ AND } \text{not}(b \text{ OR } c)$

Draw the **simplest** circuit equivalent to this truth table – the best answer has only 2 gates.

a	b	c	Output Value
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Write a simpler Boolean expression equivalent to : **not ( not A and not B ) or not C**

If you make a truth table for a circuit with 4 inputs, how many rows will it have?

By any appropriate means, show that these two expressions are **logically equivalent**:

$$(p \text{ NAND } q) \text{ AND } (p \text{ NOR } q) == p \text{ NOR } q$$

Construct a circuit for a **half-adder**, containing one **AND** gate and one **XOR** gate.

## 4.2 Boolean Logic

### Long Question

A school wants to **automate** the lights in the classrooms. Sometimes teachers forget to turn the lights off at night, and the janitor must walk around the building checking. This takes about 30 minutes each night. A consultant suggests that the lights could also be automatically controlled during the day – turned off whenever a room is not occupied or the sun is shining. This should save enough electricity to amount to 20,000 euros per year, or more.

Each room needs a motion sensor (to check whether the room is occupied), a light sensor (to sense whether it is sunny outside) and a temperature sensor (lights will be turned off if it is too warm, to save on air-conditioning costs). The logic of the system is expressed in the following algorithm:

```
boolean LightControl( boolean sunny, boolean occupied, boolean hot)
{
    if ( (sunny == false) && (occupied == true) && (hot == false) )
        return true;           // this turns the lights on
    else
        return false;         // this turns the light off
}
```

The main program will pass the values of the sensors to the appropriate parameters, and then use the function result to control the light switch.

- (a) Draw a logic circuit (using logic gates) to implement this same logic.
- (b) The control logic is equivalent to one of the following expressions:  
**not( Sunny and Hot and not Occupied)**  
**not( Sunny or Hot ) and Occupied**  
**not( Sunny or Hot and Occupied)**  
**not( Sunny or Hot and not Occupied)**  
 Choose the **simplest** equivalent expression,  
 and use a **truth table** to show that it is equivalent to the circuit from (a).
- (c) Draw a logic circuit representing your answer to (b).  
 Your circuit should contain only two logic gates.
- (d) The school wants to add a 4<sup>th</sup> component. This is a **master switch** in the central office. When the master switch is turned off, all the lights should automatically go out. When the master switch is on, then the normal logic should apply. Rewrite the original algorithm to implement the logic including the master switch.
- (e) Explain one advantage of using **logic circuit gates** rather than **algorithm logic** for controlling the lights.

## 5.2 Algorithms

For each algorithm below, answer the same three questions:

- (1) **Purpose:** Name the standard algorithm it represents (the **intended** algorithm).  
If it does **not** represent a **standard** algorithm, then **describe** the purpose of the algorithm.
- (2) Does it function correctly or is there an error? If there is an error, fix it.
- (3) If it contains **error handling code**, explain why it is necessary. If it **needs** some error handling code, describe the need and the code required.
- (4) Answer any other questions asked about the algorithm.

Assume that there is no **bad** data to cause errors. Assume that **integer** values cannot overflow, so don't state this as an error. Assume that all variables have been properly **declared**. Assume that arrays already exist. Ignore any tiny syntax errors (e.g. misspellings).

### Array Algorithms

=====

```

String[] words = new String[1000];
int size = 0;

public void ad(String data, String[] list)
{
    list[size] = data;
    size = size + 1;
}

public void fix(String[] list)
{
    int p,x;
    String t;
    for(p = 0; p < size; p = p+1)
    {
        for(x = 0; x < size; x = x+1)
            if (list[x].compareTo(list[x+1]) > 0)
            {
                t = list[x];
                list[x] = list[x+1];
                list[x+1] = t;
            }
    }
}

public int get(String item, String[] list, int a, int b)
{
    if (b < a || a < 0 || b >= size) return -1;

    int found = -1;
    int m = (a + b) / 2;

    if (list[m].equals(item))
        found = m;
    else if ( item.compareTo(list[m])>0 )
        found = get(item,list,m+1,b);
    else
        found = get(item,list,a,m-1);
    return found;
}

```

\*\*\*\*\* Write a procedure find the LARGEST VALUE in a list of numbers. \*\*\*\*\*