UNIT 2

I. Make sure that you know the following words:

Digitally formatted data, to interface the computer, types of input data, to store data, on and off patterns, circuit, numeric value, binary digit, binary code, to convert, to apply, to execute sequentially, to process, time consuming, user-friendly format, computer capabilities, advanced software, accessibility, system software, application software, programming language, operating system, utility software.

II. Read and translate the text:

SOFTWARE DEVELOPMENT

Although a modern computer is capable of accepting many types of input, it can only operate on digitally formatted data, just as original computers did. Hence software must be created to interface the computer with the various types of input data. Because a computer runs on electricity, data must be stored as a series of on and off patterns. Computer circuits can be in only one or two states: either on (represented by 1) or off (represented by 0). Each numeric value is known as a binary digit (bit) and unique combinations of those two bits, are what binary code, or machine language, is called. Different patterns in binary code could then be used to represent various input characters. Once data has been converted to binary form, computers would then apply a software program (applications or apps) to the digital input data, sequentially execute the instructions, and successfully process it into information.

Writing software programs in numerical (or digital) format was an immense task for anyone. Therefore, it was necessary to develop software into a more user-friendly format. As programmers developed existing code, new computer capabilities were noticed and the demand for even more advanced software increased. It inspired programmers to develop more software. Thousands of new programs were being written as swiftly as possible and yet the demand continued to increase. Software types typically fall into 2 categories: system software and application software.

System software controls various internal computer activities. Any software that controls such activities will fall into one of three categories: programming language, operating system and utility software.

Programming languages are the various methods of writing computer instructions. The instructions adhere to a particular set of protocols for each language. through the years, more than 200 languages have been developed, some of which are quite specialized. Some of the most popular languages include BASIC, COBOL, Pascal, C, C++, Visual Basic. But regardless of which language a program was written in, a computer can only process binary code. Therefore,

each language must eventually be converted back to binary code before any instructions can be followed.

High-level languages were developed for two reasons: one – so programmers could work on different computers without having to learn a new assembly language each time, and software written on one computer could be used by another. A compiler (program translator) was used to help solve these problems by translating a program into machine language and checking the program for syntax errors.

Until 1970, IBM bundled its software with its computers, selling the hardware along with the software needed to run it. IBM began charging a separate fee for its software, thus opening a market for independent software developers to write programs that would run on IBM machines. By the time the first personal computer (PC), called the Altair hit the market in the 1975, there were many well-developed computer languages and competent programmers available to write software for the new industry.

Operating systems have become larger and more sophisticated in response to the capabilities of new hardware and other software. CP/M developed by Gary Kildall in 1973 for Intel Corporation, was the first OS that would run on PCs made by different manufacturers, and it had the largest number of programs for data and word processing and calculations. Although it was a powerful operating system, few software developers supported it, referring to write for the growing DOS-based market.

Utility software expands the performance of the operating system by adding functions that are not part of the original OS. Utilities perform troubleshooting jobs, inspecting diskettes for damage, file conversion, defragmenting, data compression and file spooling. Some utility programs, such as Symantec's Norton Utilities, even retrieve data from damaged disks. Utilities can also be used to customize the OS environment.

III. Match the terms in the left-hand column with their definitions in the right-

hand column.

1. primary	1.grow wider or bigger
2.definition	2.put a limit on
3.expand	3.at last, finally
4.consume	4.make (a book etc) from information that has been collected
5.restrict	5.chief
6.eventually	6.make it possible for
7.compile	7.use up
8.flexible	8.precise statement or explanation
9.enable	9.follow
10.track	10.easily bent

IV. Answer the following questions:

1. Why was software created? What was the reason?
2. How must data be stored in a computer?
3. What does the "binary digit" mean?
4. Why does the unceasing demand for more advanced software exist?
5. What does the notion "programming language" mean? Can you give any examples?
6. What do you know about high-level languages?
7. What is compiler used for?
8. What do you know about OS?
9. What kind of OS do you prefer to use? And why?
10. Why do we need utilities?

V. Retell the text briefly using the following expressions:

Interface the computer with input data, digitally formatted data, binary digit, to process binary code, software on and off patterns, machine language, execute instructions, time consuming, programming languages, operating system, utility software.

VI. Fill in the gaps with the missing words from the text. Mind that in each item the first letter of the word is used:

1. At first, numbers were the primary form of c… d… .
2. Software must be created to i… the computer with input d… .
3. Each numeric value is called b… d… .
4. System software c… internal computer activities.
5. Because of h… l… l … programmers could work on different computers without having to learn a new a… language each time.
6. The o… s… is a group of programs that help computer to interpret commands, p … the inputs and outputs, and manage data.
7. Utility and application software expand the p… of OS.

VII. Rearrange the words and get the right sentences:

1. states / of two / computer / can / in only / be / circuits / one.
2. a user-friendly / it / to develop / format / into / necessary / software / was.
3. it/ to develop / more / inspired / software / programmers.
1. software / computer / written / could / be / on / used / on / one / another.
2. controls / system / various / software / activities / internal.
3. instructions / are / the / programming / various / computer / methods / of / languages / writing.
4. must / each / converted / be / binary / language / to / back / code.
5. have / operating / larger / become / and / more / systems / sophisticated.

VIII. Supply the missing preposition. Refer to the text if necessary.

1. Modern computer operates … digitally formatted data.
2. A computer runs … electricity.
3. Computer circuits can be only … two states.
4. The early days of computing were restricted … science applications.
5. It increased the demand … more advanced software.
6. OS is a group of programs that help … the operation … a computer.
7. CP / M was the first OS that could run … Cs made … different manufacturers.
8. Utilities inspect diskettes … damage.

IX. There are two words given in each item. You have to explain in what way they

are similar and how they differ from each other.

1. (a) first computers, (b) modern computers;
2. (a) OS, (b) utility;
3. (a) utility, (b) apps.

X. Give the opposites of the following words:

input; quickly; modern; fill; notice; internal; specific; separate; high; independent; competent; add; insert; early; possible.

XI. Give the synonyms of the following words:

modern; accept; realize; increase; restrict; general; swiftly; various; particular; error; memorize; competent; competent; entire.

XII. Replace the italicized word combinations by appropriate ones given in the

list below. Some of them might be used more than once. Refer to a dictionary if necessary.

Appear in ; at first; contemporary; dealing with errors and problems; desire; encourage; examine carefully and select; increase; not cease; payment; quickly; tremendous; understand.

1. ***In the beginning***, computers used numbers as the primary form of input data.
2. A ***modern*** computer is capable of accepting many types of input.
3. Engineers quickly ***realized*** that writing in binary code was extremely difficult.
4. Writing software programs in numerical format was an ***immense*** task for anyone.
5. It ***inspired*** programmers to develop more software.
6. IBM began charging a separate ***fee*** for its software.

7. IBM chose DOS as its operating system, and when its open-architecture PC ***hit*** the market, programmers ***jumped at the chance*** to write application software for it.
3. Utilities perform ***troubleshooting*** jobs, inspecting diskettes for damage.

XIII. Translate into English.

1. Хотя современный компьютер способен принимать много типов входных данных, он может работать только с цифровыми данными.
2. Следовательно, программное обеспечение должно быть создано, чтобы обеспечить связь компьютера с разными типами входных данных.
3. В начале своего развития использование компьютеров ограничивалось научными и инженерными применениями.
4. Инженеры быстро поняли, что написание программы в двоичном коде было чрезвычайно трудным и утомительным делом.
5. До 1970 года IBM ориентировало программное обеспечение на различные компьютеры и продавало их в комплекте.
6. К тому моменту, когда первый персональный компьютер появился на рынке, существовали хорошо развитые компьютерные языки и грамотные программисты способные создать новые программы.
7. При включении компьютера операционная система загружается автоматически и может активизировать другие программы.
8. Утилиты расширяют возможности операционной системы, добавляя ей дополнительные функции.

XIV. Read the text, try to understand the topic and put the items in the right order. The first item and the last one are in the right position.

1. An OCR (Optical Character Recognition) system enables you to input printed documents into your computer automatically via a scanner.
   FineReader is an omnifont optical text recognition system.It means that the system recognizes texts in practically any font without prior training. The process of document can be divided into two stages:
2. The principles of IPA (Integrity, Purposefulness, Adaptivity):
   - Integrity – the object of recognition is described as a single entity by set of basic elements and their interrelations.
   - Purposefulness – recognition is a process of generation and purposeful verification of hypotheses.
   - Adaptivity – the system ability to learn and to be trained.
3. Let's take a closer look at the second of two mentioned above stages.
   FineReader OCR image processing involves analyzing the image file transmitted by the scanner (layout analysis) and recognizing each character.
4. These three principles determine the systems behavior. The system generates a hypothesis about object of recognition and then tries to find all the structural elements and their interrelations step by step. Then the program

adapts itself to the text – tunes itself - using the positive experience gained from the first confidently recognized characters. Purposeful searching and using context enable the system to recognize even torn and distorted characters.

5. The final result is the recognized text you see in the FineReader window, a text you can edit save in any convenient format.
6. These two stages are:
   Scanning. During the first stage the scanner acts as the "eye" of your computer, it looks at the image and transfers it into the computer.

   ***Recognition*. During the second stage FineRider performs the OCR image processing.**

XV. Look through the text. Make a short summary of it.

UNIX

The Unix operating system was conceived and implemented in 1969 at AT&T's Bell Laboratories in the United States by Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna. It was first released in 1971, and initially, was written entirely in assembly language, a common practice at the time. Later, in a key pioneering approach in 1973, Unix was rewritten in the programming language C by Dennis Ritchie (with exceptions to the kernel and I/O). The availability of an operating system written in a high-level language allowed easier portability to different computer platforms.

With AT&T being required to license the operating system's source code to anyone who asked (due to an earlier antitrust case forbidding them from entering the computer business), Unix grew quickly and became widely adopted by academic institutions and businesses. In 1984, AT&T divested itself of Bell Labs. Free of the legal obligation requiring free licensing, Bell Labs began selling Unix as a proprietary product.

The GNU Project, started in 1983 by Richard Stallman, had the goal of creating a "complete Unix-compatible software system" composed entirely of free software. Work began in 1984. Later, in 1985, Stallman started the Free Software Foundation and wrote the GNU General Public License (GNU GPL) in 1989. By the early 1990s, many of the programs required in an operating system (such as libraries, compilers, text editors, a Unix shell, and a windowing system) were completed, although low-level elements such as device drivers, daemons, and the kernel were stalled and incomplete.

Linus Torvalds has said that if the GNU kernel had been available at the time (1991), he would not have decided to write his own.

Although not released until 1992 due to legal complications, development of 386BSD, from which NetBSD, OpenBSD and FreeBSD descended, predated that of Linux. Linus Torvalds has said that if 386BSD had been available at the time, he probably would not have created Linux.

Torvalds began the development of the Linux kernel on MINIX, and applications written for MINIX were also used on Linux. Later, Linux matured and further Linux kernel development took place on Linux systems. GNU applications also replaced all MINIX components, because it was advantageous to use the freely available code from the GNU Project with the fledgling operating system; code licensed under the GNU GPL can be reused in other projects as long as they also are released under the same or a compatible license. Torvalds initiated a switch from his original license, which prohibited commercial redistribution, to the GNU GPL. Developers worked to integrate GNU components with the Linux kernel, making a fully functional and free operating system.

XVI. Translate the text in written form.

C# (C Sharp programming language)

C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure. C# is built on the syntax and semantics of C++, allowing C programmers to take advantage of .NET and the common language runtime. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 5.0, which was released on August 15, 2012.

The ECMA standard lists these design goals for C#:

- The C# language is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.

- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.