

## **B.Tech.**

First Semester Examination, December-2009

### **Fundamentals of Computer & Programming in 'C' (CSE-101-F)**

---

**Note :** Attempt five questions in total. Question no. 1 is compulsory. All questions carry equal marks. Attempt at least one question from Sections A, B, C, D.

**Q. 1. (a) Which of the following is not a memory?**

(i) Flash

(ii) Bubble

(iii) Triple

**Ans. (ii) Bubble.**

**Q. 1. (b) In MICR "C" stands for :**

(i) Colour

(ii) Code

(iii) Character

**Ans. (iii) Character.**

**Q. 1. (c) What is the latest version of Windows O.S.?**

**Ans. Windows 7.**

**Q. 1. (d) LSI technique was used in which generation of Computers.**

**Ans. LSI technique was used in IIIrd generation of computers.**

**Q. 1. (e) Differentiate between system software and application software.**

**Ans. System Software :** These are the inbuilt softwares which are used by system for its functions and tasks. e.g., compilers, interpreters etc.

**Application Software :** These are the softwares built by users for performing their own tasks.

**Q. 1. (f) What type of memory is a Hard Disk, primary or secondary ?**

**Ans. Hard disk is secondary memory.**

**Q. 1. (g) What is bootstrap loader ?**

**Ans. Bootstrap Loader :** Bootstrap loader stores instruction for booting. It is found in very first sector (boot sector) of drive and read as soon as system is loaded. It is brought into main memory and its instructions are executed.

**Q. 1. (h) What is full form of TCP/IP ?**

**Ans. TCP/IP :** Transmission Control Protocol/Internet Protocol.

**Q. 1. (i) What is full form of OSI ?**

**Ans. OSI :** Open System Interface.

**Q. 1. (j) Low level language programs are faster or slower in execution as compared to HLL programs.**

**Ans. Low level languages are faster than high level languages.**

**Q. 1. (k) What is firewall?**

**Ans. Firewall :** A firewall in computer terms protects our networks from untrusted networks.

**Q. 1.(l) What are basic data types supported by C language?**

**Ans.** Basic data types supported in C :

- |                   |                   |
|-------------------|-------------------|
| (i) int (integer) | (ii) real (float) |
| (iii) char        | (iv) boolean      |

**Q. 1. (m) What is type casting?**

**Ans. Type Casting :** It is automatic type conversion in C or the conversion of one type of data into another type is called type casting.

```
Example : int a, b;  
          float result;  
          a = 77;  
          b = 12;  
          result = (float) a / b  
                  Type casting
```

a & b are of integer type, so their division will also return a integer. In order to get accurate result, we converted it to float.

**Q. 1. (n) What is difference between exit and break statement in C?**

**Ans. Break :** It transfers control out of the block where it is used.

**Exit :** Exit is a function used for the termination of execution of a C program. It is a standard library function & uses the header file stdlib.h.

**Q. 1. (o) Define a dangling reference.**

**Ans. Dangling Reference :** Dangling references are those access paths that continue to exist even after the lifetime of a variable.

**Q. 1.(p) How many passes are there in C compiler?**

**Ans.** There are 2 passes in C language.

**Q. 1. (q) Write a tertiary operator available in C language.**

**Ans.** The conditional operators ? & : form a tertiary operator

exp 1 ? exp2 : exp3

If exp1 is true exp2 is executed otherwise exp3 is executed.

**Q. 1. (r) Write one homogeneous and one heterogeneous data structure available in C language.**

**Ans. Homogeneous Data Structure :** These are the data structures in which all components are of same data type. e.g., Array.

**Heterogeneous Data Structure :** These are data structures in which all components may not be of same data type :

e.g., Lists.

**Q. 1. (s) When you close a file, where is it stored in RAM or Hard Disk?**

**Ans.** When we close a file it is stored in hard disk.

## Section—(A)

**Q. 2. (a) Briefly discuss the generations of microprocessors. What was the technology used in each generation?**

**Ans. P1 (086) First-Generation Processors :** The first generation of processors represents the series of chips from Intel that were found in the first PCs. IBM, as the architect of the PC at the time, chose Intel processors and support chips to build the PC motherboard, setting a standard that would hold for many subsequent processor generations to come.

**8088 & 8086 Processors :** Intel introduced a revolutionary new processor called the 8086 back in June of 1978. The 8086 was one of the first 16-bit processor chips on the market; at the time virtually all other processors were 8-bit designs. The 8086 had 16-bit internal registers and could run a new class of software using 16-bit instructions. It also had a 16-bit external data path, which meant it could transfer data to memory 16-bits at a time.

The address bus was 20-bits wide, meaning that the 8086 could address a full 1MB( $2^{20}$ ) of memory. This was in stark contrast to most other chips of that time that had 8-bit internal registers, an 8-bit external data bus, and a 16-bit address bus allowing a maximum of only 64KB of RAM ( $2^{16}$ ).

The cost was high because the 8086 needed a 16-bit data bus rather than a less expensive 8-bit bus. Systems available at that time were 8-bit, and slow sales of the 8086 indicated to Intel that people weren't willing to pay for the extra performance of the full 16-bit design. In response, Intel introduced a kind of crippled version of the 8086, called the 8088. The 8088 essentially deleted 8 of the 16 bits on the data bus, making the 8088 an 8-bit chip as far as data input and output were concerned. However, because it retained the full 16-bit internal registers and the 20-bit address bus, the 8088 ran 16-bit software and was capable of addressing a full 1MB of RAM.

**80186 & 80188 Processors :** After Intel produced the 8086 and 8088 chips, it turned its sights toward producing a more powerful chip with an increased instruction set. The company's first efforts along this line—the 80186 and 80188—were unsuccessful. But incorporating system components into the CPU chip was an important idea for Intel because it led to faster, better chips, such as the 286.

**8087 Coprocessor :** Intel introduced the 8086 processor in 1976. The math coprocessor that was paired with the chip—the 8087—often was called the numeric data processor (NDP), the math coprocessor, or simply the math chip. The 8087 is designed to perform high-level math operations at many times the speed of the main processor. The primary advantage of using this chip is the increased execution speed in number-crunching programs, such as spreadsheet applications.

Microprocessor incorporates most or all of the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or microchip) the first microprocessors emerged in the early 1970s and were used for electronic calculators, using binary-coded decimal (BCD) arithmetic on 4-bit words. Other embedded uses of 4- and 8-bit microprocessors, such as terminals, printers, various kinds of automation etc., followed rather quickly. Affordable 8-bit microprocessors with 16-bit addressing also led to the first general purpose microcomputers in the mid-1970.

**Q. 2. (b) Explain the working of a LASER Printer.**

**Ans. LASER Printer :** It is also known as page printer. They are nonimpact printer. They print one page at a time at a very high speed of 2000 lines per minute. They are very costly economical. In these printer, an image is produced on a photosensitive surface using a laser beam of other light source. The

laser beam it is turn of an on under the control of a computer. The areas that are expose to laser attract toner, which is generally an ink power. They can produce 3,000 pages per minute.

**Q. 3. What are the functions of an Operating System? Compare the characteristics of DOS and UNIX.**

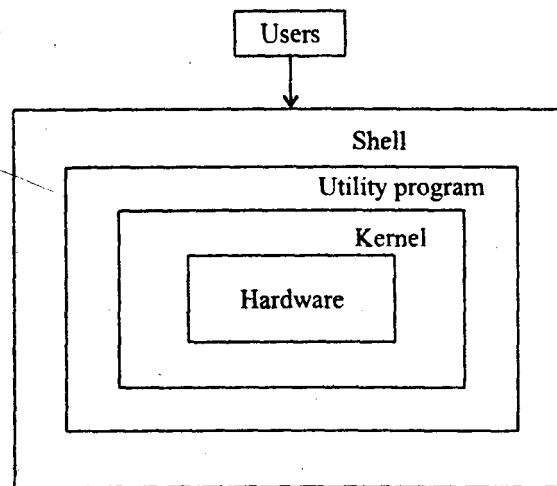
**Ans. Functions of Operating System :**

- (i) Starting the computer
- (ii) Automatic sequencing of jobs
- (iii) Performing I/O operations and handling of interrupts.
- (iv) Handling of errors in case of abnormal termination.
- (v) Scheduling of jobs.
- (vi) Allocation and control of resources.
- (vii) Provide protection to jobs and to O.S.
- (viii) Provide user friendly interface.
- (ix) Perform accounting of resources used by the user program.
- (x) Memory management
- (xi) Process management
- (xii) File management
- (xiii) Input-output device management

**Differences between UNIX and DOS :**

**(i) Unix Architecture :** It consists of 3 main parts :

- (a) Kernel
- (b) Shell
- (c) Set of utility programs



**Major Features of UNIX :**

- (i) Portability
- (ii) Multi-user operation

(iii) Multi-tasking

(iv) Device independence

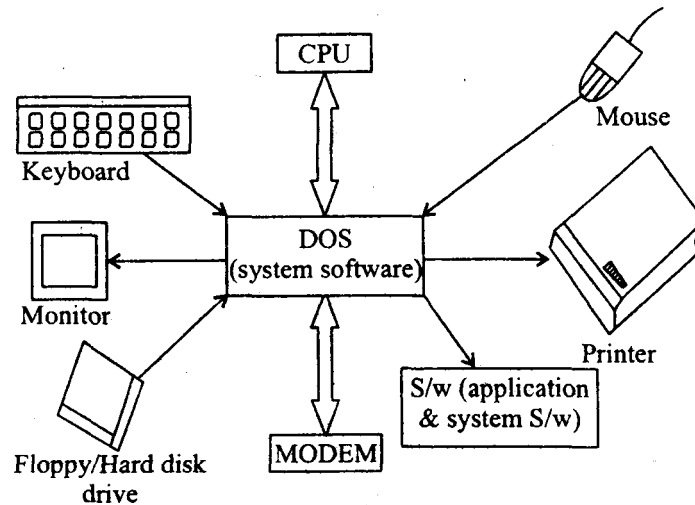
(v) Hierarchical file system

(vi) Security

(vii) Tools and tool-building utilities

(ii) **DOS** : A Disk Operating System (DOS) is a program that links the computer hardware with software and acts as a resource-manager for proper functioning of each and every part of PC.

**Architecture :**



DOS doesn't offer multi-programming and multi-processing. It is a single program OS w/c manages the various resources. It provides us with a working environment through w/c we can control the information movement. MS-DOS helps us to use applications, create and manage files, use peripherals such as printers, disks and hard disks.

### Section—(B)

**Q. 4. (a) Write the advantages, disadvantages of HLL, Low Level Language & Assembly Language.**

**Ans.** In high level languages we can write programs in English like manner and is more convenient to use. Programmer can perform complex task by using high level languages with less efforts. There are different high level languages such as FORTRAN (Formula Instructions), BASIC (Beginners All Purpose Symbolic Instruction Code), COBOL (Common Business Oriented Language), PASCAL, C++, Visual Basic, VB.Net etc.

High level languages have many advantages. High level languages are very similar to nature language such as English so they are very easy to learn and use. For higher level languages programmers needs not to learn about internal structure of the computer. High level language programs require less time and efforts.

High level programs are very easy to maintain than lower level languages. In lower level languages instruction are difficult and very hard to locate, correct and modify but in high level languages it is very easy to understand and modify when desired. In high level languages programmer needs not to

write all steps because computer take cares of all small error. Compilers of high level languages automatically catch and point out the errors made by the programmers.

The main advantage : it can directly communicate with hardware. The main disadvantage : it's very hard to write in terms of machine instructions.

Assembly language programming is one of the most powerful methods of programming a computer. It works one step above the level of the computer. That is, instead of writing a program that says "print this character on the screen when such and such a condition is met" you write a program that says "put the number \$A6 at address \$E510 when the flag at \$1023 becomes a one." Assembly programs generally use numbers in base 16, also known as hexadecimal, due to the fact that the physical structure of most computers is based on grouping the individual switches in groups of 8. Thus, it is very easy to go from the switch patterns to the hexadecimal numbers.

The commands are very short, and the computer does exactly what you tell it. The advantage to this is that the executables are extremely small and fast. To give you an idea of the speed gains, an assembly program will generally run 2 to 4 times faster than a comparable C/C++ program, and orders of magnitude faster than many other languages.

One of the disadvantage of assembly is, the computer does exactly what you tell it. That is, if you think you're telling it one thing, and you're actually telling it another, the results are going to be quite different than what you expect. Usually, they're dramatically different, and nine times out of ten they involve an infinite loop somewhere you didn't expect it.

This brings us to the second disadvantage. Due to the short and direct nature of the syntax, debugging your own program can be difficult. Even trying to figure out what someone else's program is attempting to do can take so much time that it's often easier to start from scratch. And the slightest mistake will mean the program won't work properly. Only very rarely will it even do anything close. This is simply because of the direct nature of the commands. The only way around this is to document the program far more than one would deem necessary; every line if possible.

Thus, assembly language can be considered an extremely fast, powerful, and efficient language. But one should be alert for the potential difficulties assembly language can present.

**Q. 4. (b) What is the function of a Loader and a Linker? How they are different from each other?**

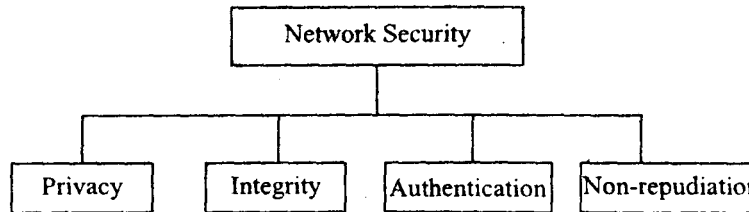
**Ans.** Linker is a tool that merges the object files produced by separate compilation or assembly and creates an executable file. Three tasks of linker—Searches the program to find library routines used by program, e.g., printf(), math routines....—Determine the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references

—Resolves references among files.

A loader is a system program that performs the loading function. Some loader also support relocation and linking others have a separate linker and loader. Basic Functions of loader-bringing an object program into memory.

**Q. 5. (a) What are different issues related with Network Security? Briefly discuss some.**

**Ans. Network Security :** Network security is becoming more and more crucial as a result of increasing volume of data and number of internet user all over the world. Different issues related to network security are :



**Privacy :** It means that the sender and the receiver aspect confidentially. The transmitted message should be received by the intended receiver.

**Integrity :** It means that the data must be received exactly as it was sent by the receiver, i.e., there must be no change during the transmission either intentionally or accidentally.

**Authentication :** It means the receiver must sure about the sender's identity.

**Non-repudiation :** It means that a receiver must be able to prove that the received message came from a specific sender. The sender must not be able to deny, what he/she did.

So, network security measure are needed to protect data during their transmission and must guarantee that the data transmission are authentic.

**Q. 5. (b) How internet can be useful for society? What are the drawbacks of net addiction?**

**Ans.** A network is a collection of autonomous computers which are connected, which are accessed by multiple users in order to exchange information.

**(i) Benefits of In'ternet :** (i) Large amount of data can be exchanged.

(ii) Less time is required.

(iii) Efficient

(iv) Communication can be developed easily between people who are living in distant geographic areas.

(v) Any information can easily be accessed about any topic very easily and less time.

**(ii) Drawbacks of Net Addiction :** (i) Cyber-crime

(ii) Increased pressure on eyes which leads to reduced eye-sight.

(iii) Mental blockage as any person who is net-addicted will not use his brain.

(iv) Reduced physical activities.

(v) No time for other works.

### Section—(C)

**Q. 6. Explain with syntax and proper mechanisms provided for loops in C.**

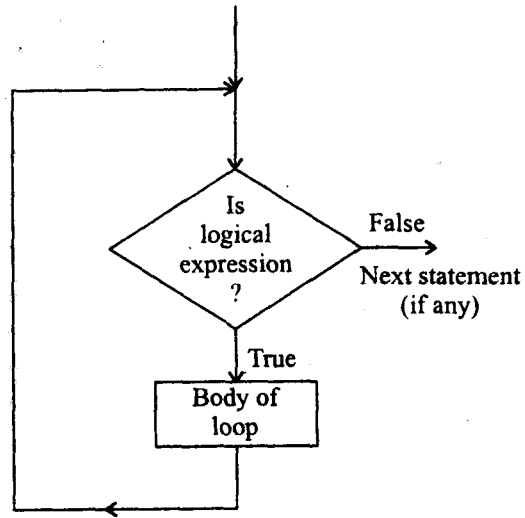
**Ans. Loops in C :** It provides us 3 loop structures— while, do-while, for

**While :**

**Syntax :** while (condition)

{

body of loop

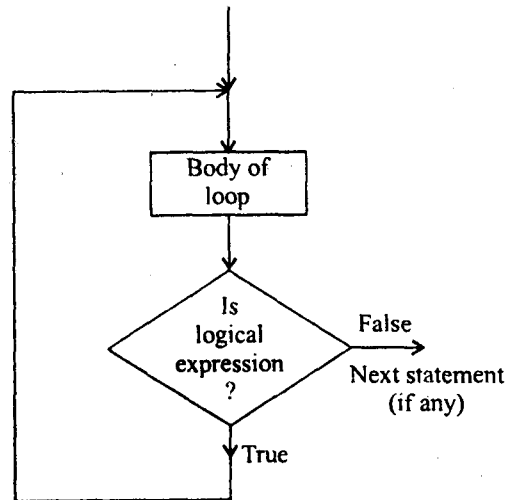


**Do-while :**

**Syntax :** do

```

{
    body of loop
}
while (condition);
  
```



**For :**

**Syntax :** for (initialization; test expression; re-initialization)

```

{
    body of loop
}
  
```



**Q. 7. WAP in C to accept the following information of 50 students :**

**Name (First, Moddle, Last)**

**Age**

**Roll No.**

**Date of Birth (DD—MM—YY)**

**& display this information on screen under proper headings.**

**Ans. Program in C :**

```
# include <stdio.h>
# include <conio.h>
# include <string.h>
void main( )
{
    char f[100], m[100], r[100];
    int a, roll, dd, mm, yy, i;
    for (i = 0; i < 50; i++)
    {
        printf("Enter your first, middle & last name");
        scanf("%s%s%s", &f, &m, &r);
        printf("Enter your age");
        scanf("%d", &a);
        printf("Enter your roll no.");
        scanf("%d", &roll);
        printf("Enter your date of birth : DDMMYY");
        scanf("%d%d%d", &dd, &mm, &yy)
    }
}
```

### **Section—(D)**

**Q. 8. (a) Using pointers WAP in C to find the desired element in an array of N elements.**

```
Ans. # include <stdio.h>
      # include <conio.h>
      # define size 20
      void main( )
      {
          int*search (int*, int);
          int a[size], e, *ptr, *temp;
          int i;
          clrscr( );
```

```

ptr = NULL;
temp = NULL;
printf("Enter the 20 element of array n");
for (i = 0; i < size; i++);
{
    scanf("%d", &a[i]);
}
printf("Enter the element to be match");
scanf("%d", &e);
ptr = search a,e;
{
    printf("The element found at %dth position", ptr + 1);
    getch( );
}
int*search (int*arr, int.el)
{
    while(*arr)
    {
        if(el == *arr)
        return arr;
        arr++;
    }
    return(null);
}

```

**Q. 8. (b) What are the disadvantages of pointer variables?**

**Ans. The Main Advantages of Using Pointers are :**

- (i) Function cannot return more than one value. But when the same function can modify many pointer variables and function as if it is returning more than one variable.
- (ii) In the case of arrays, we can decide the size of the array at runtime by allocating the necessary space.
- (iii) In the case of pointers to classes, we can use polymorphism and virtual classes to change the behaviour of pointers to various types of classes at runtime.

**Coming to the Disadvantages of Pointers :**

- (i) If sufficient memory is not available during runtime for the storage of pointers, the program may crash (least possible).
- (ii) If the programmer is not careful and consistent with the use of pointers, the program may crash (very possible).

### **A brief Synopsis on the "Rules" of Pointers :**

- (i) Always initialize a pointer by calling an allocator.
- (ii) Always check to see if the allocation request failed.
- (iii) Always deallocate memory controlled by a pointer when done with it.
- (iv) Never access memory that has not been allocated—pay attention to the size of arrays.
- (v) Never access memory that has been deallocated.
- (vi) Do not allocate and deallocate memory with wild abandon, because that can fragment the virtual memory address space, causing future allocation requests to fail—particularly in long running programs such as web servers.

### **Q. 9. (a) What are the typical error situations, which can occur while dealing with Files in C?**

**Ans. The Types of Compilation Errors :** First, let's distinguish between the types of errors : most compilers will give three types of compile-time alerts : compiler warnings, compiler errors, and linker errors.

Although you don't want to ignore them, compiler warnings aren't something severe enough to actually keep your program from compiling. Usually, compiler warnings are an indication that something might go wrong at runtime. How can the compiler know this at all? You might be making a typical mistake that the compiler knows about. A common example is using the assignment operator ('=') instead of the equality operator ('==') inside an if statement. Your compiler may also warn you about using variables that haven't been initialized and other similar mistakes. Generally, you can set the warning level of your compiler—I like to keep it at its highest level so that my compiler warnings don't turn into bugs in the running program ('runtime bugs').

Nevertheless, compiler warnings aren't going to stop you from getting your program working (unless you tell your compiler to treat warnings as errors), so they're probably a bit less frustrating than errors. Errors are conditions that prevent the compiler from completing the compilation of your files. Compiler errors are restricted to single source code files and are the result of 'syntax errors'. What this really means is that you've done something that the compiler cannot understand. For instance, the statement "for(;" isn't correct syntax because a for loop always needs to have three parts. Although the compiler would have expected a semicolon, it would also have expected a conditional expression, so the error message you get might be something like "line 53, unexpected parenthesis)". Note, also, that compiler errors will always include a line number at which the error was detected.

Even if you make it through the compilation process successfully, you may run into linker errors. Linker errors, unlike compiler errors, have nothing to do with incorrect syntax. Instead, linker errors are usually problems with finding the definitions for functions, structs, classes, or global variables that were declared, but never actually defined, in a source code file. Generally, these errors will be of the form "could not find definition for X".

Usually, the compilation process will begin with a series of compiler errors and warnings and, once you're fixed all of them, you'll then be faced with any linker errors. In turn, I'll first cover dealing with compiler errors and then with linker errors.

**Compiler Errors—Where do you Start? :** A single error near the top of your program can cause a cascade of other compiler errors because those lines might rely on something early in the program that

the compiler couldn't understand. For instance, if you declare a variable with improper syntax, the compiler will complain about the syntax error and that it cannot find a declaration for the variable. Leaving off a semicolon in the wrong place can result in an astonishing number of errors. Things like this can happen because C and C++ syntax allows for things like declaring of a type immediately after the type definition :

```
struct
{
    int x;
    int y;
} myStruct;
```

This would create a variable, myStruct, with room to store a struct containing two integers. Unfortunately, this means that if you leave off a semicolon, the compiler will interpret it as though the next thing in the program is intended to be a struct (or return a struct). Something like this

```
struct MyStructType
{
    int x;
    int y;
}
int foo( )
{ }
```

can result in an surprising number of errors (possibly including a complaint about an extraneous "int" being ignored). All this for a single character! best to start at the top.

This brings up another guiding principle of hunting down compiler errors : when in doubt, look earlier in the program. Since syntax errors can have mysterious repercussions later, it's possible that the compiler was giving a line number that doesn't actually have a syntax error! Worse, many times, the compiler won't be as friendly in telling you exactly what happened earlier in the program. Even the first compiler error you get might be due to something several lines before the indicated warning.