**C programming-answers**

**1.  What is C language?**

The C programming language is a standardized programming language developed in the early 1970s by Ken Thompson and Dennis Ritchie for use on the UNIX operating system. It has since spread to many other operating systems, and is one of the most widely used programming languages. C is prized for its efficiency, and is the most popular programming language for writing system software, though it is also used for writing applications.

**2.  What does static variable mean?**

There are 3 main uses for the static.

1. If you declare within a function: It retains the value between function calls

2. If it is declared for a function name: By default function is extern..so it will be visible from other files if the function declaration is as static..it is invisible for the outer files

3. Static for global variables: By default we can use the global variables from outside files If it is static global..that variable is limited to with in the file.

#include   
int t = 10;    
main(){  
int x = 0;   
void funct1();  
funct1();              
printf("After first call \n");  
funct1();              
printf("After second call \n");  
funct1();              
printf("After third call \n");  
}  
void funct1()  
{  
    static int y = 0;    
    int z = 10;               
    printf("value of y %d z %d",y,z);  
    y=y+10;  
}

value of y 0 z 10 After first call  
value of y 10 z 10 After second call  
value of y 20 z 10 After third call

**3.  What are the different storage classes in C?**

C has three types of storage: automatic, static and allocated.  Variable having block scope and without static specifier have automatic storage duration.

Variables with block scope, and with static specifier have static scope. Global variables (i.e, file scope) with or without the the static specifier also have static scope.  Memory obtained from calls to malloc(), alloc() or realloc() belongs to allocated storage class.

**4.  What is hashing?**

To hash means to grind up, and thats essentially what hashing is all about. The heart of a hashing algorithm is a hash function that takes your nice, neat data and grinds it into some random-looking integer.

The idea behind hashing is that some data either has no inherent ordering (such as images) or is expensive to compare (such as images). If the data has no inherent ordering, you cant perform comparison searches.

**5.  Can static variables be declared in a header file?**

You cant declare a static variable without defining it as well (this is because the storage class modifiers static and extern are mutually exclusive). A static variable can be defined in a header file, but this would cause each source file that included the header file to have its own private copy of the variable, which is probably not what was intended.

**6.  Can a variable be both constant and volatile?**

Yes. The const modifier means that this code cannot change the value of the variable, but that does not mean that the value cannot be changed by means outside this code.

The function itself did not change the value of the timer, so it was declared const. However, the value was changed by hardware on the computer, so it was declared volatile. If a variable is both const and volatile, the two modifiers can appear in either order.

**7.  Can include files be nested?**

Yes. Include files can be nested any number of times. As long as you use precautionary measures, you can avoid including the same file twice. In the past, nesting header files was seen as bad programming practice, because it complicates the dependency tracking function of the MAKE program and thus slows down compilation. Many of todays popular compilers make up for this difficulty by implementing a concept called precompiled headers, in which all headers and associated dependencies are stored in a precompiled state.

**8.  What is a null pointer?**

There are times when its necessary to have a pointer that doesnt point to anything. The macro NULL, defined in , has a value thats guaranteed to be different from any valid pointer. NULL is a literal zero, possibly cast to void\* or char\*.

Some people, notably C++ programmers, prefer to use 0 rather than NULL.   
The null pointer is used in three ways:  
1) To stop indirection in a recursive data structure.  
2) As an error value.  
3) As a sentinel value.

**9.  What is the output of printf("%d") ?**

When we write printf("%d",x); this means compiler will print the value of x. But as here, there is nothing after %d so compiler will show in output window garbage value.

**10.  What is the difference between calloc() and malloc() ?**

calloc(...) allocates a block of memory for an array of elements of a certain size. By default the block is initialized to 0. The total number of memory allocated will be (number\_of\_elements \* size).

malloc(...) takes in only a single argument which is the memory required in bytes. malloc(...) allocated bytes of memory and not blocks of memory like calloc(...).

malloc(...) allocates memory blocks and returns a void pointer to the allocated space, or NULL if there is insufficient memory available.

calloc(...) allocates an array in memory with elements initialized to 0 and returns a pointer to the allocated space. calloc(...) calls malloc(...) in order to use the C++ \_set\_new\_mode function to set the new handler mode.

**11.  What is the difference between printf() and sprintf() ?**

sprintf() writes data to the character array whereas printf(...) writes data to the standard output device.

**12.  How to reduce a final size of executable?**

Size of the final executable can be reduced using dynamic linking for libraries.

**13.  Can you tell me how to check whether a linked list is circular?**

Create two pointers, and set both to the start of the list. Update each as follows:

while (pointer1) {  
pointer1 = pointer1->next;  
pointer2 = pointer2->next;   
if (pointer2) pointer2=pointer2->next;  
if (pointer1 == pointer2) {  
print ("circular");  
}  
}

If a list is circular, at some point pointer2 will wrap around and be either at the item just before pointer1, or the item before that. Either way, its either 1 or 2 jumps until they meet.

**14.  Advantages of a macro over a function?**

Macro gets to see the Compilation environment, so it can expand \_\_ \_\_TIME\_\_ \_\_FILE\_\_ #defines. It is expanded by the preprocessor.

For example, you cant do this without macros  
#define PRINT(EXPR) printf( #EXPR =%d\n, EXPR)   
PRINT( 5+6\*7 ) // expands into printf(5+6\*7=%d, 5+6\*7 );   
You can define your mini language with macros:  
#define strequal(A,B) (!strcmp(A,B))

**15.  What is the difference between strings and character arrays?**

A major difference is: string will have static storage duration, whereas as a character array will not, unless it is explicity specified by using the static keyword.

Actually, a string is a character array with following properties:   
\* the multibyte character sequence, to which we generally call string, is used to initialize an array of static storage duration. The size of this array is just sufficient to contain these characters plus the terminating NUL character.   
\* it not specified what happens if this array, i.e., string, is modified.   
\* Two strings of same value[1] may share same memory area.

**16.  Write down the equivalent pointer expression for referring the same element a[i][j][k][l] ?**

a[i] == \*(a+i)  
a[i][j] == \*(\*(a+i)+j)  
a[i][j][k] == \*(\*(\*(a+i)+j)+k)  
a[i][j][k][l] == \*(\*(\*(\*(a+i)+j)+k)+l)

**17.  Which bit wise operator is suitable for checking whether a particular bit is on or off?**

The bitwise AND operator. Here is an example:

enum {  
KBit0 = 1,  
KBit1,  
  
KBit31,  
};  
if ( some\_int & KBit24 )  
printf ( Bit number 24 is ON\n );  
else  
printf ( Bit number 24 is OFF\n );

**18.  Which bit wise operator is suitable for turning off a particular bit in a number?**

The bitwise AND operator, again. In the following code snippet, the bit number 24 is reset to zero.

some\_int = some\_int & ~KBit24;

**19.  Which bit wise operator is suitable for putting on a particular bit in a number?**

The bitwise OR operator. In the following code snippet, the bit number 24 is turned ON:  
some\_int = some\_int | KBit24;

**20. Does there exist any other function which can be used to convert an integer or a float to a string?**

Some implementations provide a nonstandard function called itoa(), which converts an integer to string.

#include   
char \*itoa(int value, char \*string, int radix);  
DESCRIPTION  
The itoa() function constructs a string representation of an integer.  
PARAMETERS  
value: Is the integer to be converted to string representation.   
string: Points to the buffer that is to hold resulting string.  
The resulting string may be as long as seventeen bytes.   
radix: Is the base of the number; must be in the range 2 - 36.   
A portable solution exists. One can use sprintf():  
char s[SOME\_CONST];  
int i = 10;  
float f = 10.20;  
sprintf ( s, %d %f\n, i, f );

**21.  Why does malloc(0) return valid memory address ? What's the use?**

malloc(0) does not return a non-NULL under every implementation. An implementation is free to behave in a manner it finds suitable, if the allocation size requested is zero. The implmentation may choose any of the following actions:

\* A null pointer is returned.  
\* The behavior is same as if a space of non-zero size was requested. In this case, the usage of return value yields to undefined-behavior.

Notice, however, that if the implementation returns a non-NULL value for a request of a zero-length space, a pointer to object of ZERO length is returned! Think, how an object of zero size should be represented

For implementations that return non-NULL values, a typical usage is as follows:  
void  
func ( void )  
{  
int \*p; /\* p is a one-dimensional array, whose size will vary during the the lifetime of the program \*/  
size\_t c;  
p = malloc(0); /\* initial allocation \*/  
if (!p)  
{  
perror (FAILURE );  
return;  
}  
/\* \*/  
while (1)  
{  
c = (size\_t) ; /\* Calculate allocation size \*/  
p = realloc ( p, c \* sizeof \*p );  
/\* use p, or break from the loop \*/  
/\* \*/  
}  
return;  
}  
Notice that this program is not portable, since an implementation is free to return NULL for a malloc(0) request, as the C Standard does not support zero-sized objects.

**22.  Difference between const char\* p and char const\* p**

In const char\* p, the character pointed by p is constant, so u cant change the value of character pointed by p but u can make p refer to some other location.

In char const\* p, the ptr p is constant not the character referenced by it, so u cant make p to reference to any other location but u can change the value of the char pointed by p.

**23.  What is the result of using Option Explicit?**

When writing your C program, you can include files in two ways. The first way is to surround the file you want to include with the angled brackets < and >. This method of inclusion tells the preprocessor to look for the file in the predefined default location. This predefined default location is often an INCLUDE environment variable that denotes the path to your include files.

For instance, given the INCLUDE variable   
INCLUDE=C:\COMPILER\INCLUDE;S:\SOURCE\HEADERS; using the #include version of file inclusion, the compiler first checks the C:\COMPILER\INCLUDE directory for the specified file. If the file is not found there, the compiler then checks the  S:\SOURCE\HEADERS directory. If the file is still not found, the preprocessor checks the current directory.

The second way to include files is to surround the file you want to include with double quotation marks. This method of inclusion tells the preprocessor to look for the file in the current directory first, then look for it in the predefined locations you have set up. Using the #include file version of file inclusion and applying it to the preceding example, the preprocessor first checks the current directory for the specified file. If the file is not found in the current directory, the C:COMPILERINCLUDE directory is searched. If the file is still not found, the preprocessor checks the S:SOURCEHEADERS directory.

The #include method of file inclusion is often used to include standard headers such as stdio.h or stdlib.h.

The #include file include nonstandard header files that you have created for use in your program. This is because these headers are often modified in the current directory, and you will want the preprocessor to use your newly modified version of the header rather than the older, unmodified version.

**24.  What is the benefit of using an enum rather than a #define constant?**

The use of an enumeration constant (enum) has many advantages over using the traditional symbolic constant style of #define. These advantages include a lower maintenance requirement, improved program readability, and better debugging capability.  
1) The first advantage is that enumerated constants are generated automatically by the compiler. Conversely, symbolic constants must be manually assigned values by the programmer.  
2) Another advantage of using the enumeration constant method is that your programs are more readable and thus can be understood better by others who might have to update your program later.

3) A third advantage to using enumeration constant

# C Basics

**C Introduction**  
  
C is a general-purpose computer programming language developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories for use with the Unix operating system. Although C was designed for implementing system software,it is also widely used for developing portable application software.C is one of the most widely used programming languages of all time and there are very few computer architectures for which a C compiler does not exist. C has greatly influenced many other popular programming languages, most notably C++, which began as an extension to C.  
  
**C History**  
  
Developed between 1969 and 1973 along with Unix  
Due mostly to Dennis Ritchie  
Designed for systems programming  
1)Operating systems  
2)Utility programs  
3)Compilers  
4)Filters  
  
Original machine (DEC PDP-11) was very small  
1)24K bytes of memory, 12K used for operating system  
2)Written when computers were big, capital equipment  
3)Group would get one, develop new language, OS  
  
**Characteristics of C**  
  
1)Small size   
2)Extensive use of function calls   
3)Loose typing -- unlike PASCAL   
4)Structured language   
5) Low level (BitWise) programming readily available   
6)Pointer implementation - extensive use of pointers for memory, array, structures and functions.   
  
**C Program Structure**  
  
A C program basically has the following form:   
1)Preprocessor Commands   
2)Type definitions   
3)Function prototypes -- declare function types and variables passed to function.   
4) Variables   
5)Functions   
  
We must have a main() function.

# C- Advanced

1.**Can we have a pointer to a function?**

a. Not possible

b. Possible

c. Depends on the return value

d. Depends on the # of arguments

*Answer:b.Possible*

void func(int a)  
{  
}  
void main()  
{  
void (\*fp)(int);  
fp=func;  
fp(1);  
}

2.**Write a function to swaps the values of two integers.**

a. No solution exist.

b. Solution need 2 extra variables

c Solution exist without any extra variables

d. Solution need 1 extra variable

*Solution c Solution exist without any extra variables*

1. void swap(int a, int b)  
{  
int c;  
c=a;  
a=b;  
b=c;  
}

2. void swap (int a, int b)  
{

a=a+b;

b=a-b;

a=a-b;

}  
*Solution2* is the best solution since no extra variable is required.

3.**Which of the following Bitwise operators can be used efficiently to swap two numbers?**

a.   &

b. ^

c. |

d. ||

*Solution:b ^*

a=a^b  
b=a^b  
a=a^b

Now ‘a’ will have ‘b’s initial value and wise-versa.

4.**Do you find any issue with the above snippet of code?**

 a.  No issues

b. P is a bad pointer

c P is a void pointer

d. Both 2& 3

*Answer: b. P is a bad pointer*

When a pointer is first allocated, it does not have a pointee. The pointer is "uninitialized" or simply "bad". A de-reference operation on a bad pointer is a serious run-time error. If you are lucky, the de-reference operation will crash or halt immediately (Java behaves this way). If you are unlucky, the bad pointer de-reference will corrupt a random area of memory, slightly altering the operation of the program so that it goes wrong some indefinite time later. Each pointer must be assigned a pointee before it can support de-reference operations. Before that, the pointer is bad and must not be used. Bad pointers are very common. In fact, **every pointer starts out with a bad value**. Correct code overwrites the bad value with a correct reference to a pointee, and thereafter the pointer works fine. There is nothing automatic that gives a pointer a valid pointee.

void BadPointer()  
{

int\* p;            // allocate the pointer, but not the pointee

\*p = 42;        // this de-reference is a serious runtime error

}  
  
5.**void add(int a, int b)**

**{**

**int c;**

**c =  a + b;**

**add (1,1);**

**}**

**What is the result of above funtion?**

a. Sum of a,b,1

b. Results in Buffer Overflow

c Results in Compiler Error

d Results in Stack Overflow

*Answer: d. Results in Stack Overflow*

When a function is called recursively, sometimes infinite recursions occur which results in STACK OVERFLOW. What does it mean? Well when a function is called,

1. First it will evaluate actual parameter expressions.

2. Then, memory is allocated to local variables.

3. Store caller’s current address of execution (return address of the current function) and then continue execute the recursive call.

4. Then it executes rest of the function body and reaches end and returns to the caller’s address.

Now when a function is infinitely called (recursively) without a proper condition to check its recursive, then only first 3 steps keep executing and function will never reach step 4 to finish execution and return to previous function. In this way, function will keep allocating memory and at some point of time it will go out of memory or reaches stack limit and will never be able to accommodate another function and hence crashes. This is called stack overflow.

6.**Which of the following will initialize the new memory to 0 ?**

a. malloc

b.     free

c.   new

d    delete

*Answer: c. new*

“new” will initlize the new memory to 0 but “malloc()” gives random value in the new alloted memory location

7.**Which of the following standard C library converts a string to a long integer and reports any .leftover. numbers that could not be converted.**

a.   atol

b.   atoi

c.  stol

d.  strtol

*Answer: d. strtol*

strtol() Converts a string to a long integer and reports any .leftover. numbers that could not be converted.  
atoi() Converts a string to an integer.  
atol() Converts a string to a long integer.

**C Interview Questions and Answers**

**1)  How do you construct an increment statement or decrement statement in C?**

Answer:There are actually two ways you can do this. One is to use the increment operator ++ and decrement operator . For example, the statement x++ means to increment the value of x by 1. Likewise, the statement x means to decrement the value of x by 1. Another way of writing increment statements is to use the conventional + plus sign or minus sign. In the case of x++, another way to write it is x = x +1?.

**2) Some coders debug their programs by placing comment symbols on some codes instead of deleting it. How does this aid in debugging?**

Answer:Placing comment symbols /\* \*/ around a code, also referred to as commenting out, is a way of isolating some codes that you think maybe causing errors in the program, without deleting the code. The idea is that if the code is in fact correct, you simply remove the comment symbols and continue on. It also saves you time and effort on having to retype the codes if you have deleted it in the first place.

**3) What is the equivalent code of the following statement in WHILE LOOP format?**  
[c]  
for (a=1; a<=100; a++)  
printf ("%d\n", a \* a);  
[/c]

Answer:[c]  
a=1;  
while (a<=100) {  
printf ("%d\n", a \* a);  
a++;  
}  
[/c]

**4) What is spaghetti programming?**

Answer:Spaghetti programming refers to codes that tend to get tangled and overlapped throughout the program. This unstructured approach to coding is usually attributed to lack of experience on the part of the programmer. Spaghetti programing makes a program complex and analyzing the codes difficult, and so must be avoided as much as possible.

**5) In C programming, how do you insert quote characters ( and ) into the output screen?**

Answer:This is a common problem for beginners because quotes are normally part of a printf statement. To insert the quote character as part of the output, use the format specifiers \ (for single quote), and \ (for double quote).

**6) What is the use of a \0' character?**

Answer:It is referred to as a terminating null character, and is used primarily to show the end of a string value.

**7) What is the difference between the = symbol and == symbol?**

Answer:The = symbol is often used in mathematical operations. It is used to assign a value to a given variable. On the other hand, the == symbol, also known as equal to or equivalent to, is a relational operator that is used to compare two values.

**8) Which of the following operators is incorrect and why? ( >=, <=, <>, ==)**

Answer:<> is incorrect. While this operator is correctly interpreted as not  equal to in writing conditional statements, it is not the proper operator to be used in C programming. Instead, the operator  !=  must be used to indicate not equal to condition.

**9) Can the curly brackets { } be used to enclose a single line of code?**

Answer:While curly brackets are mainly used to group several lines of codes, it will still work without error if you used it for a single line. Some programmers prefer this method as a way of organizing codes to make it look clearer, especially in conditional statements.

**10) What are header files and what are its uses in C programming?**

Answer:Header files are also known as library files. They contain two essential things: the definitions and prototypes of functions being used in a program. Simply put, commands that you use in C programming are actually functions that are defined from within each header files. Each header file contains a set of functions. For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

**11) Can I use  int data type to store the value 32768? Why?**

Answer:No. int data type is capable of storing values from -32768 to 32767. To store 32768, you can use long int instead. You can also use unsigned int, assuming you dont intend to store negative values.

**12) Can two or more operators such as \n and \t be combined in a single line of program code**

Answer:Yes, its perfectly valid to combine operators, especially if the need arises. For example: you can have a code like printf (Hello\n\n\World\) to output the text Hello on the first line and World enclosed in single quotes to appear on the next two lines.

**13) Why is it that not all header files are declared in every C program?**

Answer:The choice of declaring a header file at the top of each C program would depend on what commands/functions you will be using in that program. Since each header file contains different function definitions and prototype, you would be using only those header files that would contain the functions you will need. Declaring all header files in every program would only increase the overall file size and load of the program, and is not considered a good programming style.

**14) When is the void keyword used in a function?**

Answer:When declaring functions, you will decide whether that function would be returning a value or not. If that function will not return a value, such as when the purpose of a function is to display some outputs on the screen, then void is to be placed at the leftmost part of the function header. When a return value is expected after the function execution, the data type of the return value is placed instead of void.

**15) What are compound statements?**

Answer:Compound statements are made up of two or more program statements that are executed together. This usually occurs while handling conditions wherein a series of statements are executed when a TRUE or FALSE is evaluated. Compound statements can also be executed within a loop. Curly brackets { } are placed before and after compound statements.

**16) Write a loop statement that will show the following output:**  
1  
12  
123  
1234  
12345  
Answer:[c]  
for (a=1; a<=5; i++) {  
for (b=1; b<=a; b++)  
printf("%d",b);  
printf("\n");  
}  
[/c]

**17) What is wrong in this statement?  scanf(%d,whatnumber);**

Answer:An ampersand & symbol must be placed before the variable name whatnumber. Placing & means whatever integer value is entered by the user is stored at the address of the variable name. This is a common mistake for programmers, often leading to logical errors.

**18) How do you generate random numbers in C?**

Answer:Random numbers are generated in C using the rand() command. For example: anyNum = rand() will generate any integer number beginning from 0, assuming that anyNum is a variable of type integer.

**19) What could possibly be the problem if a valid function name such as tolower() is being reported by the C compiler as undefined?**  
The most probable reason behind this error is that the header file for that function was not indicated at the top of the program. Header files contain the definition and prototype for functions and commands used in a C program. In the case of tolower(), the code #include  must be present at the beginning of the program.

**20) What does the format %10.2 mean when included in a printf statement?**

Answer:This format is used for two things: to set the number of spaces allotted for the output number and to set the number of decimal places. The number before the decimal point is for the allotted space, in this case it would allot 10 spaces for the output number. If the number of space occupied by the output number is less than 10, addition space characters will be inserted before the actual output number. The number after the decimal point sets the number of decimal places, in this case, its 2 decimal spaces.

**21) What is wrong with this statement? myName = Robin;**

Answer:You cannot use the = sign to assign values to a string variable. Instead, use the strcpy function. The correct statement would be: strcpy(myName, Robin);

**22) How do you determine the length of a string value that was stored in a variable?**

Answer:To get the length of a string value, use the function strlen(). For example, if you have a variable named FullName, you can get the length of the stored string value by using this statement: I = strlen(FullName); the variable I will now have the character length of the string value.

**23) Is it possible to initialize a variable at the time it was declared?**

Answer:Yes, you dont have to write a separate assignment statement after the variable declaration, unless you plan to change it later on.  For example: char planet[15] = Earth; does two things: it declares a string variable named planet, then initializes it with the value Earth.

**24) What are the different file extensions involved when programming in C?**

Answer:Source codes in C are saved with .C file extension. Header files or library files have the .H file extension. Every time a program source code is successfully compiled, it creates an .OBJ object file, and an executable .EXE file.

**25) What are reserved words?**

Answer:Reserved words are words that are part of the standard C language library. This means that reserved words have special meaning and therefore cannot be used for purposes other than what it is originally intended for. Examples of reserved words are int, void, and return.

**26) What are linked list?**

Answer:A linked list is composed of nodes that are connected with another. In C programming, linked lists are created using pointers. Using linked lists is one efficient way of utilizing memory for storage.

**27) What are binary trees?**

Answer:Binary trees are actually an extension of the concept of linked lists. A binary tree has two pointers, a left one and a right one. Each side can further branch to form additional nodes, which each node having two pointers as well.

**28) Not all reserved words are written in lowercase. TRUE or FALSE?**

Answer:FALSE. All reserved words must be written in lowercase; otherwise the C compiler would interpret this as unidentified and invalid.

**29) What is wrong with this program statement? void = 10;**

Answer:The word void is a reserved word in C language. You cannot use reserved words as a user-defined variable.

**30) Is this program statement valid? INT = 10.50;**

Answer:Assuming that INT is a variable of type float, this statement is valid. One may think that INT is a reserved word and must not be used for other purposes. However, recall that reserved words are express in lowercase, so the C compiler will not interpret this as a reserved word.

**31) What is a newline escape sequence?**

Answer:A newline escape sequence is represented by the \n character. This is used to insert a new line when displaying data in the output screen. More spaces can be added by inserting more \n characters. For example, \n\n would insert two spaces. A newline escape sequence can be placed before the actual output expression or after.

**32) What is output redirection?**

Answer:It is the process of transferring data to an alternative output source other than the display screen. Output redirection allows a program to have its output saved to a file. For example, if you have a program named COMPUTE, typing this on the command line as COMPUTE >DATA can accept input from the user, perform certain computations, then have the output redirected to a file named DATA, instead of showing it on the screen.

**33) What is the difference between functions abs() and fabs()?**

Answer:These 2 functions basically perform the same action, which is to get the absolute value of the given value. Abs() is used for integer values, while fabs() is used for floating type numbers. Also, the prototype for abs() is under , while fabs() is under .

**34) Write a simple code fragment that will check if a number is positive or negative.**

Answer:[c]

If (num>=0)  
printf("number is positive");  
else  
printf ("number is negative");  
[/c]

**35) What does the function toupper() do?**

Answer:It is used to convert any letter to its upper case mode. Toupper() function prototype is declared in . Note that this function will only convert a single character, and not an entire string.

**36) Which function in C can be used to append a string to another string?**

Answer:The strcat function. It takes two parameters, the source string and the string value to be appended to the source string.

**37) Dothese two program statements perform the same output? 1) scanf(%c, &letter);  2) letter=getchar()**

Answer:Yes, they both do the exact same thing, which is to accept the next key pressed by the user and assign it to variable named letter.

**38) What is the difference between text files and binary files?**

Answer:Text files contain data that can easily be understood by humans. It includes letters, numbers and other characters. On the other hand, binary files contain 1s and 0s that only computers can interpret.

**39) is it possible to create your own header files?**

Answer:Yes, it is possible to create a customized header file. Just include in it the function prototypes that you want to use in your program, and use the #include directive followed by the name of your header file.

**40) What is dynamic data structure?**

Answer:Dynamic data structure provides a means for storing data more efficiently into memory. Using dynamic memory allocation, your program will access memory spaces as needed. This is in contrast to static data structure, wherein the programmer has to indicate a fix number of memory space to be used in the program.

**41) The % symbol has a special use in a printf statement. How would you place this character as part of the output on the screen?**

Answer:You can do this by using %% in the printf statement. For example, you can write printf(10%%) to have the output appear as 10% on the screen.

**42) What are the advantages and disadvantages of a heap?**

Answer:Storing data on the heap is slower than it would take when using the stack. However, the main advantage of using the heap is its flexibility. Thats because memory in this structure can be allocated and remove in any particular order. Slowness in the heap can be compensated if an algorithm was well designed and implemented.