[**Turbo c interiew questions with answers**](http://c-interviews-questions.blogspot.com/)

[C Interview Questions Part 2](http://c-interviews-questions.blogspot.com/2009/09/qwhat-will-be-output-of-following-code.html)

**Q:What will be the output of the following code?**

 void main ()

      {      int i = 0 , a[3] ;

      a[i] = i++;

      printf (“%d",a[i]) ;

      }

**Ans**: The output for the above code would be a garbage value. In the statement a[i] = i++; the value of the variable i would get assigned first to a[i] i.e. a[0] and then the value of i would get incremented by 1. Since a[i] i.e. a[1] has not been initialized, a[i] will have a garbage value.

**Q:Why doesn't the following code give the desired result?**

**A**:  int x = 3000, y = 2000 ;

      long int z = x \* y ;

      Ans: Here the multiplication is carried out between two ints x and y, and  the result that would overflow would be truncated before being assigned to  the variable z of type long int. However, to get the correct output, we should use an explicit cast to force long arithmetic as shown below:

      long int z = ( long int ) x \* y ;

      Note that ( long int )( x \* y ) would not give the desired effect.

**Q: Why doesn't the following statement work?**

**A:**                        char str[ ] = "Hello" ;

                        strcat ( str, '!' ) ;

  Ans: The string function strcat( ) concatenates strings and not a character. The basic difference between a string and a character is that a string is a collection of characters, represented by an array of characters whereas a character is a single character. To make the above statement work writes the statement as shown below:

                        strcat ( str, "!" ) ;

**Q:How do I know how many elements an array can hold**?

**A:** The amount of memory an array can consume depends on the data type of an array. In DOS                               environment, the amount of memory an array can consume depends on the current memory model (i.e. Tiny, Small, Large, Huge, etc.). In general an array cannot consume more than 64 kb. Consider following program, which shows the maximum number of elements an array of type int, float and char can have in case of Small memory model.

                              main( )

                              {

                              int i[32767] ;

                              float f[16383] ;

                              char s[65535] ;

                              }

**Q:How do I write code that reads data at memory location specified by segment and offset?**

**A:** Use peekb( ) function. This function returns byte(s) read from specific segment and offset locations in memory. The following program illustrates use of this function. In this program from VDU memory we have read characters and its attributes of the first row. The information stored in file is then further read and displayed using peek( ) function.

#include

#include

main( )

{

char far \*scr = 0xB8000000 ;

FILE \*fp ;

int offset ;

char ch ;

if ( ( fp = fopen ( "scr.dat", "wb" ) ) == NULL )

{

printf ( "\nUnable to open file" ) ;

exit( ) ;

}

// reads and writes to file

for ( offset = 0 ; offset < 160 ; offset++ )

fprintf ( fp, "%c", peekb ( scr, offset ) ) ;

fclose ( fp ) ;

if ( ( fp = fopen ( "scr.dat", "rb" ) ) == NULL )

{

printf ( "\nUnable to open file" ) ;

exit( ) ;

}

// reads and writes to file

for ( offset = 0 ; offset < 160 ; offset++ )

{

fscanf ( fp, "%c", &ch ) ;

printf ( "%c", ch ) ;

}

fclose ( fp ) ;

}

[C Interview Questions Part 1](http://c-interviews-questions.blogspot.com/2009/09/c-interview-questions-part-1.html)

**Q: What is C language?**

**A**: 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.

**Q: Inprintf() Function What is the output of printf("%d")?**

**A**:1. When we write printf("%d",x); this means compiler will print the value of x. But as here, there is nothing after ï¿½%dï¿½ so compiler will show in output window garbage value.

2. When we use %d the compiler internally uses it to access theargument in the stack (argument stack). Ideally compiler determines the offset of the data variable depending on the format specification string. Now when we write printf("%d",a) then compiler first accesses the top most element in the argument stack of the printf which is %d and depending on the format string it calculated to offset to the actual datavariable in the memory which is to be printed. Now when only %d will be present in the printf then compiler will calculate the correct offset (which will be the offset to access the integer variable) but as the actual data object is to be printed is not present at that memory location so it will print what ever will be the contents of that memory location.

3. Some compilers check the format string and will generate an error without the proper number and type of arguments for things like printf(...) and scanf(...).

**Q:malloc() Function- What is the difference between "calloc(...)" and "malloc(...)"?**

**A**:1. 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(...).

2. 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.

**Q: In printf() Function- What is the difference between "printf(...)" and "sprintf(...)"?**

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

**Q:Compilation How to reduce a final size of executable?**

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

**Q:Linked Lists -- Can you tell me how to check whether a linked list is circular?**

**A**: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.

**Q:string Processing --- Write out a function that prints out all the permutations of a string. For example, abc would give you abc, acb, bac, bca, cab, cba.**

**A**:void PrintPermu (char \*sBegin, char\* sRest) {

int iLoop;

char cTmp;

char cFLetter[1];

char \*sNewBegin;

char \*sCur;

int iLen;

static int iCount;

iLen = strlen(sRest);

if (iLen == 2) {

iCount++;

printf("%d: %s%s\n",iCount,sBegin,sRest);

iCount++;

printf("%d: %s%c%c\n",iCount,sBegin,sRest[1],sRest[0]);

return;

} else if (iLen == 1) {

iCount++;

printf("%d: %s%s\n", iCount, sBegin, sRest);

return;

} else {

// swap the first character of sRest with each of

// the remaining chars recursively call debug print

sCur = (char\*)malloc(iLen);

sNewBegin = (char\*)malloc(iLen);

for (iLoop = 0; iLoop <>

strcpy(sCur, sRest);

strcpy(sNewBegin, sBegin);

cTmp = sCur[iLoop];

sCur[iLoop] = sCur[0];

sCur[0] = cTmp;

sprintf(cFLetter, "%c", sCur[0]);

strcat(sNewBegin, cFLetter);

debugprint(sNewBegin, sCur+1);

}

}

}

void main() {

char s[255];

char sIn[255];

printf("\nEnter a string:");

scanf("%s%\*c",sIn);

memset(s,0,255);

PrintPermu(s, sIn);

}