**Some basic questions in C**

**What is the difference between if-else & switch**

1. The else-if will go through each statement and check to see if its true. the switch goes directly to the true statement (if any).
2. When we have if else or if else tree, we have many compare instructions ( assembly generated by compiler)

whereas switch has only one compare and jump instruction. If the idea is to does something after comparing the values, it is always better to go with the switch case than if else tree.

1. In the if - else, first the condition is verified, then it comes to else whereas in the switch - case first it checks the cases and then it switches to that particular case.

**What is the difference between break & continue**

**break** - The break statement is used to jump out of loop. After the break statement control passes to the immediate statement after the loop. **continue** - Using continue we can go to the next iteration in loop. **exit** - it is used to exit the execution of program. break and continue are *statements,* exit is *function.*

Consider an example

int i;  
for (i=0;i<10;i++)  
{  
  
if (i==5)  
continue;  
printf("%d",i);  
if (i==8)  
break;  
}

This code will print 1 to 8 except 5.  
  
Continue means, whatever code that follows the continue statement WITHIN the loop code block will not be exectued and the program will go to the next iteration, in this case, when the program reaches i=5 it checks the condition in the if statement and executes 'continue', everything after continue, which are the printf statement, the next if statement, will not be executed.  
  
Break statement will just stop execution of the look and go to the next statement after the loop if any. In this case when i=8 the program will jump out of the loop. Meaning, it wont continue till i=9,10.  
  
break is also used in switch-case statements to delimit the different cases.

Note:Break and continue can only be applicable to loops.

**What is the Difference between while/do-while/for loops**

* **for loop** is called a determinate loop meaning that we usually use it when we know ahead how many times iterations we need. **while loop** is called a indeterminate loop because we normally use the while loop when we dont know the number of times the loop may iterate. In **do while** the statements are executed for the first time and then the conditions are tested, if the condition turns out to be true then the statements are executed again.
* The main difference comes into picture when you use continue with them i.e. for and while.   
  In a while loop if continue is used before the increment of a variable is done it converts into a infinite loop.   
  Therefore it will not turn into a infinite loop.

i=1;   
while(i<10)   
{   
/\* do stuff \*/   
if(i==6);   
continue;   
i++;   
}   
The above piece of code will turn into an infinite loop.   
for(i=1;i<10;i++)   
{   
/\* do stuff \*/   
if(i==6);   
continue;   
}

In the above for loop the value of will be incremented once it attains the value 6.

**What are the size of different data types**

sizeof(int) = 4 sizeof(p),\*p is int pointer) = 4

sizeof(unsigned int) = 4 sizeof(\*p),\*p is int pointer) = 4

sizeof(char) = 1 sizeof(q),\*q is char pointer) = 4

sizeof(unsigned char) = 1 sizeof(\*q),\*q is char pointer) = 1

sizeof(float) = 4 sizeof(a),\*a is long double pointer) = 4

sizeof(short) = 2 sizeof(\*a),\*a is long double pointer) = 12

sizeof(short int) = 2

sizeof(long) = 4

sizeof(long int) = 4

sizeof(double) = 8

sizeof(long double) = 12

**What is goto**

The goto statement transfers program execution to some label within the program.

goto skip\_point;

printf("This part was skipped.\n");

skip\_point:

printf("Hi there!\n");

**Note**: goto cannot be used outside the corresponding function where it is used

**Which is faster i++ or i = i + 1??**

**i=i+1:**

To perform this operation compiler has to perform two operations:

(1)ADD      \\i+1  
(2)Assignment operation          \i=x  
  
**i++:**

To perform this operation compiler has to perform only one operation:

(1)INR  
  
Note: above operation are microprocessor instructions i.e. in assembly language

so n++ is faster

**What printf & scanf returns**

Upon a successful return, the printf() function returns the number of characters printed (not including the trailing '\0' used to end output to strings).

printf("%d",(printf("Hello"))); Result is Hello5

scanf returns??

**What is recursive functions in C**

Recursive function is a function which contains a call to itself. Recursive function allows you to divide your complex problem into identical single simple cases which can handle easily. This is also a well-known computer programming technique: divide and conquer. Recursive function must have at least one exit condition that can be satisfied. Otherwise, the recursive function will call itself repeatly until the runtime stack overflows.

void recurse()

{

recurse(); //Function calls itself

}

int main()

{

recurse(); //Sets off the recursion

}

**What is re-entrant function?**

A function that can be safely called while it's already executing is said to be *re-entran*t. There are two occasions in which this scenario can happen: when the function is invoked by two or more distinct threads of the same program, or when executing a signal handler that raises the same signal once again. In either case, remember that calling a non re-entrant function while it's already executing is a bad idea — the program's behavior is undefined in this case.

**What is the difference between reentrant and recursive?**

**What are the Operations on Pointers are valid**

**Boundary checking in array**

**Array initialization**