Pointers Quiz

Name: 50

Q1/What is the stack?
And what is the output from this code?

```
#include "stdio.h"
int* func ()
{
int a = 5 ;
int *pa = &a ;
return pa ;
}
int main ()
{
int x = 10 ;
int* ptr ;
ptr = func ();
printf ("%d", *ptr+x);
return 0 ;
}
```

5

Q2/ write C Code to Count number of bits to be flipped to convert A to B ?
The same question bin another shape
Write C code to determine how many different bits between two integers?

Example :
Input : a = 10, b = 20
Output : 4
Binary representation of a is 00001010
Binary representation of b is 00010100
We need to flip highlighted four bits in a to make it b.

Input : a = 7, b = 10
Output : 3
Binary representation of a is 00000111
Binary representation of b is 00001010
We need to flip highlighted three bits in a to make it b.

5

Q3/ Complete the following statement

7

global variables -------> data memory
static variables -------> ……
constant data types -----> ……,
local variables(declared and defined in functions) --------> ……
variables declared and defined in main function -----> …..
pointers(ex: char *arr, int *arr) -------> ……..
dynamically allocated space(using malloc, calloc, realloc) --------> ……..

Q4/Define the Following C Keywords ?
#define?? #include?? #pragma?? #asm?? #ifdef...#endif?
o #define ……
o #include ……
#pragma ……
o #asm ……
o #ifdef ……
o #endif ……

5

Q5/Using the variable a, give definitions for the following:

a) An integer
b) A pointer to an integer
c) A pointer to a pointer to an integer
d) An array of 10 integers
e) An array of 10 pointers to integers
f) A pointer to an array of 10 integers
g) A pointer to a function that takes an integer as an argument and returns an integer
h) An array of ten pointers to functions that take an integer argument and return an integer

8

Q6/Embedded systems are often characterized by requiring the programmer to access a specific memory location. On a certain project it is required to set an integer variable at the absolute address 0x67a9 to the value 0xaa55. The compiler is a pure ANSI compiler. Write code to accomplish this task

3

Q7/what is the output

```
/*
 * main.c
 *
 *  Created on: Jun 16, 2017
 *      Author: Keroles Shenouda
 */

#include<stdio.h>
void fun(int arr[])
{
  int i;
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  for (i = 0; i < arr_size; i++)
      printf("\n %d ", arr[i]);
}
```

```
int main()
{
  int i;
  int arr[4] = {10, 20 ,30, 40};
  int arr_size =
sizeof(arr)/sizeof(arr[0]);
  for (i = 0; i < arr_size; i++)
      printf("%d ", arr[i]);
  fun(arr);
  return 0;
}
```

8

Q8/ For (intelligent Embedded Engineers)

Write a Best C Code to implement this example by Using **Pointers**, **Union** and **Bitfields** in **C** programming and can access address "`0x00008000`"

1. individual bits of a memory location (1 Byte).

or

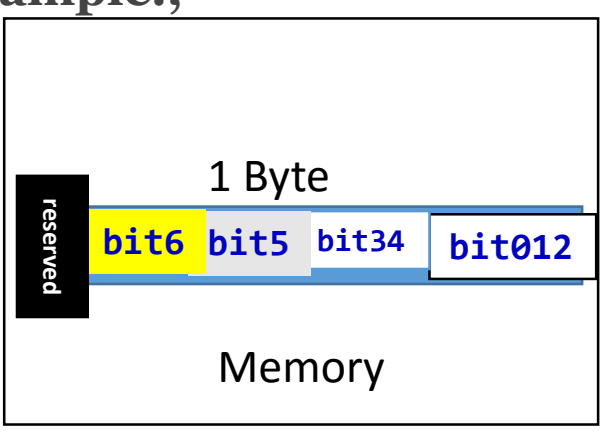
2. Read/write on all the memory block (1 byte)

For example.,

10

bit7

`0x00008000`

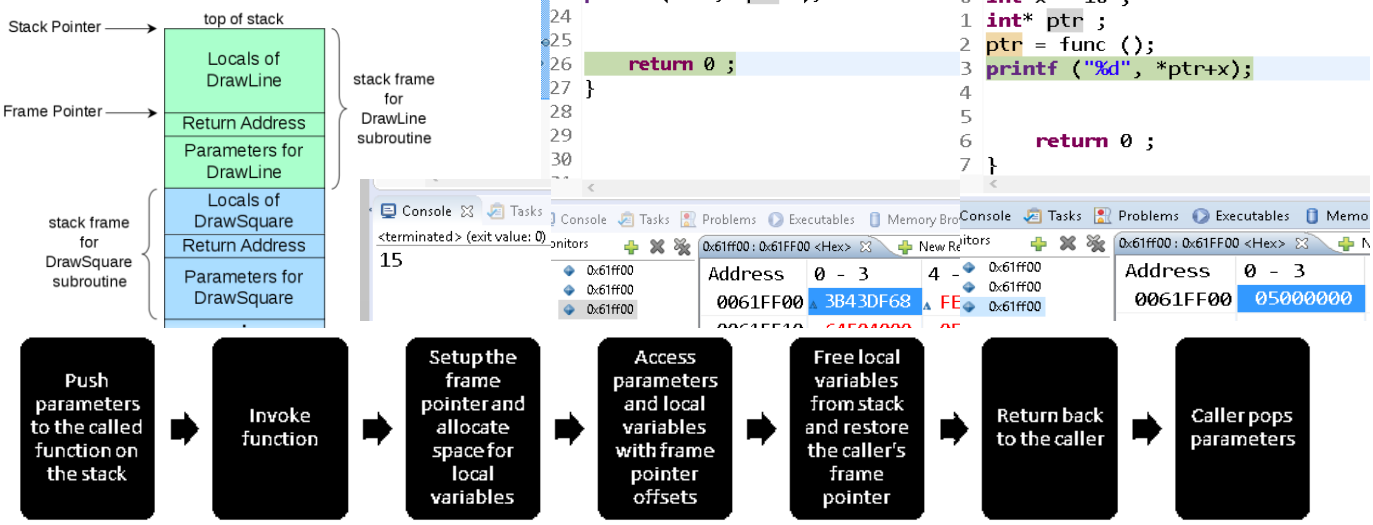


~~Wish for it~~

Work for it

Pointers Quiz Solution

Q1/The Stack



What is the stack? It's a special region of your computer's memory that stores temporary variables created by each function (including the main() function). The stack is a "LIFO" (last in, first out) data structure, that is managed and optimized by the CPU quite closely. Every time a function declares a new variable, it is "pushed" onto the stack. **Then every time a function exits, all of the variables pushed onto the stack by that function, are freed (that is to say, they are deleted).** Once a stack variable is freed, that region of memory becomes available for other stack variables.
The advantage of using the stack to store variables, is that memory is managed for you. You don't have to allocate memory by hand,

---

Q2/

```
// Count number of bits to be flipped
// to covert A into B
#include <iostream>
using namespace std;

// Function that count set bits
int countSetBits(int n)
{
   int count = 0;
   while (n)
   {
      count += n & 1;
      n >>= 1;
   }
   return count;
}

// Function that return count of
// flipped number
int FlippedCount(int a, int b)
{
   // Return count of set bits in
   // a XOR b
   return countSetBits(a^b);
}

// Driver code
int main()
{
   int a = 10;
   int b = 20;
   cout << FlippedCount(a, b)<<endl;
   return 0;
}
```

Q3/ Complete the following statement

global variables -------> data memory
static variables -------> data
constant data types -----> (depends on the compiler).For example, in the GCC compiler, on most machines, read-only variables, constants, and jump tables are placed in the **text section.**
local variables(declared and defined in functions) --------> stack
variables declared and defined in main function -----> stack
pointers(ex: char *arr, int *arr) -------> heap or data or stack, depending on the context. C lets you declare a global or a static pointer, in which case the pointer itself would end up in the data segment.
dynamically allocated space(using malloc, calloc, realloc) --------> heap

---

Q4/Define the Following C Keywords ?

#define?? #include?? #pragma?? #asm?? #ifdef...#endif?
o #define define a macro
o #include include a source file
o #pragma is for compiler directives that are machine specific or operating system specific, i.e. it tells the compiler to do something, set some
option, take some action, override some default, etc. that may or may not apply
to all machines and operating systems.
o #asm to include the assembly directive
o #ifdef compile these lines if NAME is not defined
o #endif to delimit the scope of these directives

Q5/
The answers are:
a) int a; // An integer
b) int *a; // A pointer to an integer
c) int **a; // A pointer to a pointer to an integer
d) int a[10]; // An array of 10 integers
e) int *a[10]; // An array of 10 pointers to integers
f) int (*a)[10]; // A pointer to an array of 10 integers
g) int (*a)(int); // A pointer to a function a that takes an integer argument and returns an integer
h) int (*a[10])(int); // An array of 10 pointers to functions that take an integer argument and return an integer

---

Q6/
```
int *ptr;
ptr = (int *)0x67a9;
*ptr = 0xaa55;
```
A more obscure approach is:
```
*(int * const)(0x67a9) = 0xaa55;
```

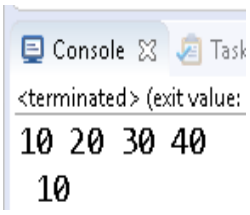
Q8/
```
/*
 * main.c
 *
 *  Created on: Jun 16, 2017
 *      Author: Keroles Shenouda
 */

#include "stdio.h"

struct Sbits {
unsigned char bit012:3;
unsigned char bit34:2;
unsigned char bit5:1;
unsigned char bit6:1;
unsigned char bit7:1;
} status;
union registerType
{
unsigned char Byte;
struct Sbits bits ;

};
int main ()
{
// define a pointer and cast it to point to the registers memory location
union registerType *pReg = (union registerType*)0x00008000;
// use the fields as
pReg->bits.bit5 = 1;
pReg->bits.bit012 = 7;
// access the whole byte as
pReg->Byte = 0x55;

return 0 ;
}
```



Q7/

Explanation:
The first printf will print 10 20 30 40
The second printf will print 10

The Explanation
In C, array parameters are always treated as pointers. So following two statements have the same meaning.

void fun(int arr[])
void fun(int *arr)
[] is used to make it clear that the function expects an array, it doesn't change anything though. People use it only for readability so that the reader is clear about the intended parameter type. The bottom line is, sizeof should never be used for array parameters, a separate parameter for array size (or length) should be passed to fun(). So, in the given program, arr_size contains ration of pointer size and integer size, this ration= is compiler dependent.