Cradle Of Bytes

Practice Questions

```
/*  Prepared By Cradle of Bytes Team, Teqnix 2013
 *  Licensed under Creative Commons
 */
```

1..

```
void main()

{

int i=32767;

printf("%d",++i);

}
```

ANS == -32768

2..

```
main()
{
        float me = 1.1;
        double you = 1.1;
        if(me==you)
                printf("I love U");
        else
           printf("I hate U");
}
```

ANS= I hate U (Note: even if me == 1.1, ans will be same)

3..

```
main()
{
        int i=-1,j=-1,k=0,l=2,m;
        m=i++&&j++&&k++||l++;
```

```
        printf("%d %d %d %d %d",i,j,k,l,m);
}
```

ANS == 0 0 1 3 1

4..

```
main()
{
        int i=5;
        printf("%d %d %d %d %d %d", i++ ,i-- ,++i ,--i ,i );
}
```

ANS == 4 5 5 4 5

5..

```
#define square(x) x*x
main()
{
        int i;
        i = 64/square(4);
        printf("%d",i);
}
```

ANS = 64

6. Consider the function

```
        find(int x, int y){
          return(( x < y ) ? 0 : ( x -y ));
        }
```

Let a,b be two non-negative integers. The call find(a, find(a,b)) can be used to find the

(a) maximum of a, b (b) positive difference of a, b

(c) sum of a, b (d) minimum of a, b

7..

```
#define f(g,g2) g##g2
main()
{
        int var12=100;
```

```
        printf("%d",f(var,12));
}
```

ANS == 100 (NOTE: ## is a token pasting operator. When f(var, 12) is executed,
the macro converts into var##12, ultimately leading to var12.
Thus, the printf() statement will now become printf("%d", var12);
and that will print 100 on screen.)

8..
The expected output of the following C program is to print the elements in the array. But when actually run, it doesn't do so.

```
 #include<stdio.h>

 #define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))
 int array[] = {23,34,12,17,204,99,16};

 int main()
 {
    int d;

    for(d=-1;d <= (TOTAL_ELEMENTS-2);d++)
       printf("%d\n",array[d+1]);

    return 0;
 }
```

Find out what's going wrong.

9..What is the output of

```
  printf("Hello, Contestents %d",printf("Welcome to Cradle of Bytes. "));
```

(a) Welcome to Cradle of Bytes. Hello, Contestents 27
(b) Hello, Contestents. Welcome to Cradle of Bytes. 20
(c) Welcome to Cradle of Bytes. Hello, Contestents 20
(d) Hello, Contestents. Welcome to Cradle of Bytes. 27

10..
What's the expected output for the following program and why?

```
enum {false,true};
int main()
{
      int i=1;
      do
```

```
        {
                printf("%d\n",i);
                i++;
                if(i < 15)
                        continue;
        }while(false);
        return 0;
}
```

11..

The following program doesn't "seem" to print "hello-out". (Try executing it)

```
 #include <stdio.h>
 #include <unistd.h>
 int main()
 {
        while(1)
        {
                fprintf(stdout,"hello-out");
                fprintf(stderr,"hello-err");
                sleep(1);
        }
        return 0;
 }
```

What could be the reason?

12..

```
#include <stdio.h>
 #define f(a,b) a##b
 #define g(a)   #a
 #define h(a) g(a)

 int main()
 {
        printf("%s\n",h(f(1,2)));
        printf("%s\n",g(f(1,2)));
        return 0;
 }
```

Just by looking at the program one "might" expect the output to be, the same for both the printf statements. But on running the program you get it as:

```
bash$ ./a.out
12
f(1,2)
bash$
```

Why is it so?

13..

```
#include<stdio.h>
 int main()
 {
        int a=10;
        switch(a)
        {
              case '1':
                 printf("ONE\n");
                 break;
              case '2':
                 printf("TWO\n");
                 break;
              defa1ut:
                 printf("NONE\n");
        }
        return 0;
 }
```

If you expect the output of the above program to be NONE, I would request you to check it out!!

14..

The following C program segfaults of IA-64, but works fine on IA-32.

```
 int main()
 {
    int* p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
    return 0;
 }
```

Why does it happen so?

15..

Here is a small piece of program(again just 14 lines of program) which counts the number of bits set in a number.

Input
Output
0
0(0000000)
5
2(0000101)
7
3(0000111)

```
 int CountBits (unsigned int x )
 {
```

```
    static unsigned int mask[] = { 0x55555555,
       0x33333333,
       0x0F0F0F0F,
       0x00FF00FF,
       0x0000FFFF16..
       } ;

       int i ;
       int shift ; /* Number of positions to shift to right*/
       for ( i =0, shift =1; i < 5; i ++, shift *= 2)
             x = (x & mask[i ])+ ( ( x >> shift) & mask[i]);
       return x;
 }
```

Find out the logic used in the above program.

16..

What do you think would be the output of the following program and why? (If you are about to say "f is 1.0", I would say check it out again)

```
#include <stdio.h>

int main()
{
      float f=0.0f;
      int i;

      for(i=0;i<10;i++)
            f = f + 0.1f;

      if(f == 1.0f)
            printf("f is 1.0 \n");
      else
            printf("f is NOT 1.0\n");

      return 0;
}
```

17..

I thought the following C program is perfectly valid (after reading about the comma operator in C). But there is a mistake in the following program, can you identify it?

```
#include <stdio.h>

int main()
{
      int a = 1,2;
      printf("a : %d\n",a);
      return 0;
```

```
}
```

What would be the output of the following C program? (Is it a valid C program?)

```
#include <stdio.h>
int main()
{
       int i=43;
       printf("%d\n",printf("%d",printf("%d",i)));
       return 0;
}
```

```
  void duff(register char *to, register char *from, register int count)
  {
     register int n=(count+7)/8;
     switch(count%8){
     case 0: do{ *to++ = *from++;
     case 7:  *to++ = *from++;
     case 6: *to++ = *from++;
     case 5: *to++ = *from++;
     case 4: *to++ = *from++;
     case 3: *to++ = *from++;
     case 2: *to++ = *from++;
     case 1: *to++ = *from++;
             }while( --n >0);
     }
  }
```

Is the above valid C code? If so, what is it trying to acheive and why would anyone do something like the above?

18..

Here is yet another implementation of CountBits. Verify whether it is correct (how do you that???). If so, find out the logic used.

```
  int CountBits(unsigned int x)
  {
     int count=0;
     while(x)
     {
        count++;
        x = x&(x-1);
     }
     return count;
  }
```

19..

Are the following two function prototypes same?

```
  int foobar(void);
```

```
int foobar();
```

20..

The following programs should be of some help in finding the answer: (Compile and run both the programs and see what happens)
Program 1:

```
#include <stdio.h>
void foobar1(void)
{
 printf("In foobar1\n");
}

void foobar2()
{
 printf("In foobar2\n");
}

int main()
{
   char ch = 'a';
   foobar1();
   foobar2(33, ch);
   return 0;
}
```

Program 2:

```
#include <stdio.h>
void foobar1(void)
{
 printf("In foobar1\n");
}

void foobar2()
{
 printf("In foobar2\n");
}

int main()
{
   char ch = 'a';
   foobar1(33, ch);
   foobar2();
   return 0;
}
```

21..

What's the output of the following program and why?

```
#include <stdio.h>
int main()
{
 float a = 12.5;
 printf("%d\n", a);
 printf("%d\n", *(int *)&a);
 return 0;
}
```

22..

The following is a small C program split across files. What do you expect the output to be, when both of them compiled together and run?

File1.c

```
int arr[80];
```

File2.c

```
extern int *arr;
int main()
{
   arr[1] = 100;
   return 0;
}
```

23..

Explain the output of the following C program (No, the output is not 20).

```
#include<stdio.h>
int main()
{
   int a=1;
   switch(a)
   {   int b=20;
      case 1: printf("b is %d\n",b);
            break;
      default:printf("b is %d\n",b);
            break;
   }
   return 0;
}
```

24..

What is the output of the following program? (Again, it is not 40, (if the size of integer is 4)).

```
#define SIZE 10
void size(int arr[SIZE])
```

```
{
        printf("size of array is:%d\n",sizeof(arr));
}

int main()
{
        int arr[SIZE];
        size(arr);
        return 0;
}
```

25..

The following is a simple c program, in which there is a function called Error to display errors. Can you see a potential problem with the way Error is defined?

```
#include <stdlib.h>
#include <stdio.h>
void Error(char* s)
{
   printf(s);
   return;
}

int main()
{
   int *p;
   p = malloc(sizeof(int));
   if(p == NULL)
   {
      Error("Could not allocate the memory\n");
      Error("Quitting....\n");
      exit(1);
   }
   else
   {
      /*some stuff to use p*/
   }
   return 0;
}
```

26..

What is the differnce between the following function calls to scanf?(Please notice the space carefully in the second call. Try removing it and observe the behaviour of the program)

```
#include <stdio.h>
int main()
{
   char c;
   scanf("%c",&c);
```

```
    printf("%c\n",c);

    scanf(" %c",&c);
    printf("%c\n",c);

    return 0;
}
```

27..

What is the potential problem with the following C program?

```
#include <stdio.h>
int main()
{
    char str[80];
    printf("Enter the string:");
    scanf("%s",str);
    printf("You entered:%s\n",str);

    return 0;
}
```

28..

What is the output of the following program?

```
#include <stdio.h>
int main()
{
    int i;
    i = 10;
    printf("i : %d\n",i);
    printf("sizeof(i++) is: %d\n",sizeof(i++));
    printf("i : %d\n",i);
    return 0;
}
```

29..

Why does the following program give a warning? (Please remember that sending a normal pointer to a function requiring const pointer does not give any warning)

```
#include <stdio.h>
void foo(const char **p) { }
int main(int argc, char **argv)
{
        foo(argv);
        return 0;
}
```

30..

What is the output of the following program?

```
#include <stdio.h>
int main()
{
        int i;
        i = 1,2,3;
        printf("i:%d\n",i);
        return 0;
}
```

31..

The following is a piece of code which implements the reverse Polish Calculator. There is a(are) serious(s) bug in the code. Find it(them) out!!! Assume that the function getop returns the appropriate return values for operands, opcodes, EOF etc..

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 80
#define NUMBER '0'

int getop(char[]);
void push(double);
double pop(void);
int main()
{
    int type;
    char s[MAX];

    while((type = getop(s)) != EOF)
    {
        switch(type)
        {
            case NUMBER:
                push(atof(s));
                break;
            case '+':
                push(pop() + pop());
                break;
            case '*':
                push(pop() * pop());
                break;
            case '-':
                push(pop() - pop());
```

```
                break;
            case '/':
                push(pop() / pop());
                break;
            /*  ...
             *  ...
             *  ...
             */
        }
    }
}
```

32..

The following is a simple program which implements a minimal version of banner command available on most *nix systems. Find out the logic used in the program.

```
#include<stdio.h>
#include<ctype.h>

char t[]={
    0,0,0,0,0,0,12,18,33,63,
    33,33,62,32,62,33,33,62,30,33,
    32,32,33,30,62,33,33,33,33,62,
    63,32,62,32,32,63,63,32,62,32,
    32,32,30,33,32,39,33,30,33,33,
    63,33,33,33,4,4,4,4,4,4,
    1,1,1,1,33,30,33,34,60,36,
    34,33,32,32,32,32,32,63,33,51,
    45,33,33,33,33,49,41,37,35,33,
    30,33,33,33,33,30,62,33,33,62,
    32,32,30,33,33,37,34,29,62,33,
    33,62,34,33,30,32,30,1,33,30,
    31,4,4,4,4,4,33,33,33,33,
    33,30,33,33,33,33,18,12,33,33,
    33,45,51,33,33,18,12,12,18,33,
    17,10,4,4,4,4,63,2,4,8,
    16,63
    };

int main(int argc,char** argv)
{

    int r,pr;
    for(r=0;r<6;++r)
        {
        char *p=argv[1];

        while(pr&&*p)
```

```
            {
            int o=(toupper(*p++)-'A')*6+6+r;
            o=(o<0||o>=sizeof(t))?0:o;
            for(pr=5;pr>=-1;--pr)
                {
                printf("%c",( ( (pr>=0) && (t[o]&(1<<pr)))?'#':' '));

                }
            }
        printf("\n");
        }
    return 0;
}
```

33..

What is the output of the following program?

```
#include <stdio.h>
#include <stdlib.h>

#define SIZEOF(arr) (sizeof(arr)/sizeof(arr[0]))

#define PrintInt(expr) printf("%s:%d\n",#expr,(expr))
int main()
{
    /* The powers of 10 */
    int pot[] = {
        0001,
        0010,
        0100,
        1000
    };
    int i;

    for(i=0;i<SIZEOF(pot);i++)
        PrintInt(pot[i]);
    return 0;
}
```

34..

The following is the implementation of the Euclid's algorithm for finding the G.C.D(Greatest Common divisor) of two integers. Explain the logic for the below implementation and think of any possible improvements on the current implementation.
BTW, what does scanf function return?

```
#include <stdio.h>
int gcd(int u,int v)
```

```
{
   int t;
   while(v > 0)
   {
      if(u > v)
      {
         t = u;
         u = v;
         v = t;
      }
      v = v-u;
   }
   return u;
}

int main()
{
   int x,y;
   printf("Enter x y to find their gcd:");
   while(scanf("%d%d",&x, &y) != EOF)
   {
      if(x >0 && y>0)
         printf("%d %d %d\n",x,y,gcd(x,y));
            printf("Enter x y to find their gcd:");
   }
   printf("\n");
   return 0;
}
```

Also implement a C function similar to the above to find the GCD of 4 integers.

35..

What's the output of the following program. (No, it's not 10!!!)

```
#include <stdio.h>
#define PrintInt(expr) printf("%s : %d\n",#expr,(expr))
int main()
{
   int y = 100;
   int *p;
   p = malloc(sizeof(int));
   *p = 10;
   y = y/*p; /*dividing y by *p */;
   PrintInt(y);
   return 0;
}
```

36..

The following is a simple C program to read a date and print the date. Run it and explain the behaviour

```
#include <stdio.h>
int main()
{
   int day,month,year;
   printf("Enter the date (dd-mm-yyyy) format including -'s:");
   scanf("%d-%d-%d",&day,&month,&year);
   printf("The date you have entered is %d-%d-%d\n",day,month,year);
   return 0;
}
```

37..

The following is a simple C program to read and print an integer. But it is not working properly. What is(are) the mistake(s)?

```
#include <stdio.h>
int main()
{
   int n;
   printf("Enter a number:\n");
   scanf("%d\n",n);

   printf("You entered %d \n",n);
   return 0;
}
```

38..

The following is a simple C program which tries to multiply an integer by 5 using the bitwise operations. But it doesn't do so. Explain the reason for the wrong behaviour of the program.

```
#include <stdio.h>
#define PrintInt(expr) printf("%s : %d\n",#expr,(expr))
int FiveTimes(int a)
{
   int t;
   t = a<<2 + a;
   return t;
}

int main()
{
   int a = 1, b = 2,c = 3;
   PrintInt(FiveTimes(a));
   PrintInt(FiveTimes(b));
   PrintInt(FiveTimes(c));
```

```
    return 0;
}
```

39..

Is the following a valid C program?

```
#include <stdio.h>
#define PrintInt(expr) printf("%s : %d\n",#expr,(expr))
int max(int x, int y)
{
   (x > y) ? return x : return y;
}

int main()
{
   int a = 10, b = 20;
   PrintInt(a);
   PrintInt(b);
   PrintInt(max(a,b));
}
```

40..

The following is a piece of C code, whose intention was to print a minus sign 20 times. But you can notice that, it doesn't work.

```
#include <stdio.h>
int main()
{
   int i;
   int n = 20;
   for( i = 0; i < n; i-- )
      printf("-");
   return 0;
}
```

Well fixing the above code is straight-forward. To make the problem interesting, you have to fix the above code, by changing exactly one character. There are three known solutions. See if you can get all those three.

41..

What's the mistake in the following code?

```
#include <stdio.h>
int main()
{
   int* ptr1,ptr2;
   ptr1 = malloc(sizeof(int));
```

```
    ptr2 = ptr1;
    *ptr2 = 10;
    return 0;
}
```

42..

What is the output of the following program?

```
#include <stdio.h>
int main()
{
    int cnt = 5, a;

    do {
        a /= cnt;
    } while (cnt --);

    printf ("%d\n", a);
    return 0;
}
```

43..

What is the output of the following program?

```
#include <stdio.h>
int main()
{
    int i = 6;
    if( ((++i < 7) && ( i++/6)) || (++i <= 9))
        ;
    printf("%d\n",i);
    return 0;
}
```

44..

What is the bug in the following program?

```
#include <stdlib.h>
#include <stdio.h>
#define SIZE 15
int main()
{
    int *a, i;

    a = malloc(SIZE*sizeof(int));
```

```
    for (i=0; i<SIZE; i++)
        *(a + i) = i * i;
    for (i=0; i<SIZE; i++)
        printf("%d\n", *a++);
    free(a);
    return 0;
}
```

45..

Is the following a valid C program? If so, what is the output of it?

```
#include <stdio.h>
int main()
{
  int a=3, b = 5;

  printf(&a["Ya!Hello! how is this? %s\n"], &b["junk/super"]);
  printf(&a["WHAT%c%c%c  %c%c  %c !\n"], 1["this"],
     2["beauty"],0["tool"],0["is"],3["sensitive"],4["CCCCCC"]);
  return 0;
}
```

46..

What is the output of the following, if the input provided is:
Life is beautiful

```
#include <stdio.h>
int main()
{
    char dummy[80];
    printf("Enter a string:\n");
    scanf("%[^a]",dummy);
    printf("%s\n",dummy);
    return 0;
}
```

47..

Note : This question has more to do with Linker than C language
We have three files a.c, b.c and main.c respectively as follows:
a.c

```
int a;
```
b.c

```
int a = 10;
```
main.c

```
extern int a;
int main()
{
	printf("a = %d\n",a);
	return 0;
}
```
Let's see what happens, when the files are compiled together:
```
bash$ gcc a.c b.c main.c
bash$ ./a.out
a = 10
```
Hmm!! no compilation/linker error!!! Why is it so??

48..

The following is the offset macros which is used many a times. Figure out what is it trying to do and what is the advantage of using it.
```
  #define offsetof(a,b) ((int)(&(((a*)(0))->b)))
```

49..

The following is the macro implementation of the famous, Triple xor swap.
```
  #define SWAP(a,b) ((a) ^= (b) ^= (a) ^= (b))
```
What are the potential problems with the above macro?

50..

What is the use of the following macro?
```
  #define DPRINTF(x) printf("%s:%d\n",#x,x)
```