Problem Solving with C

Quiz #1

Compiled by

M S Anand (anandms@pes.edu)

Text Book(s):

1. "How To Solve It By Computer", R G Dromey, Pearson, 2011.
2. "The C Programming Language", Brian Kernighan, Dennis Ritchie, 2nd Edition, Prentice Hall PTR, 1988.

Reference Book(s):

1. "Expert C Programming; Deep C secrets", Peter van der Linden
2. " The C puzzle Book", Alan R Feuer

What is the output of this program?

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

Output: 1
The above program prints 1. Associativity of comma operator is from left to right, but = operator has higher precedence than comma operator. Therefore the statement i = 1, 2, 3 is treated as (i = 1), 2, 3 by the compiler.

What is the output of this program?

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

    getchar();
     return 0;
}
```

Output is i = 3

```
#include <stdio.h>
int main(void)
{
    int first = 50, second = 60, third;
    third = first /* Will this comment work? */ + second;
    printf("%d /* And this? */ \n", third);

    getchar();
    return 0;
}
```

Output: 110 /* And this? */

Explanation: Compiler removes everything between “/*” and “*/” if they are not present inside double quotes (“”).

What is the output of this program?

```
#include <stdio.h>
int main(void)
{
    int c = 5, no = 1000;
    do {
        no /= c;
    } while(c--);
    printf ("%d\n", no);
    return 0;
}
```

Output: Exception – Divide by zero

Explanation: There is a bug in the above program. It goes inside the do-while loop for c = 0 also. Be careful when you are using do-while loop like this!!

What is the output of this program?

```
#include<stdio.h>
int main(void)
{
    char c = 'A'+256;
    printf("%c", c);
    return;
}
```

A – A
B – B
C - Overflow error at runtime
D - Compile error

Answer : A
Explanation
A, the range of ASCII values for the ASCII characters is 0-255. Hence the addition operation circulates back to ‘A’

What is the output of the following program?

```
#include<stdio.h>
int main(void)
{
   int i = 1;
   while(i++<=5);
       printf("%d ",i++);
    return 0;
}
```

A – 4
B – 7
C - 2 6
D - 2 4

Answer - B

What is the output of the below code snippet.

```
#include<stdio.h>
int main(void)
{
   printf("%d", -11%2);
   return 0;
}
```

A – 1
B - -1
C - 5.5
D - -5.5

```
#include<stdio.h>

int main(void)
{
   char s1[50], s2[50] = "Hello";
   s1 = s2;
   printf("%s", s1);
   return 0;
}
```

A – Hello
B - No output
C - Compile error
D - Runtime error

What will be the output of the following C code?

```
#include <stdio.h>

int main(void)
{
    int a[3] = {1, 2, 3};
    int *p = a;

    printf("%p\t%p", p, a);
    return 0;

}
```

a) Same address is printed
b) Different addresses are printed
c) Compile time error
d) Nothing

What will be the output of the following C code?

```
#include <stdio.h>
int main(void)
{

    char *s= "hello";
    char *p = s;

    printf("%c\t%c", p[0], s[1]);
    return 0;
}
```

a) Run time error
b) h h
c) h e
d) h l

What will be the output of the following C code?

```
#include <stdio.h>
int main(void)

{
    char *s= "hello";
    char *p = s;

    printf("%c\t%c", *(p + 3),  s[1]);
    return 0;
}
```

a) h e
b) l l
c) l o
d) l e

```
int main(void)
{
    int x,y=2,z,a;
    if ( x = y%2)
         z =2;
    a=2;
    printf("%d %d ",z,x);
    return 0;
}
```

Output:

```
< some garbage value of z > 0
```

Explanation:

This question has some stuff for operator precedence. If the condition of if is met, then z will be initialized to 2 otherwise z will contain garbage value. But the condition of if has two operators: assignment operator and modulus operator. The precedence of modulus is higher than assignment. So y%2 is zero and it'll be assigned to x. So the value of x becomes zero which is also the effective condition for if. And therefore, condition of if is false.

```
int main(void)
{
    int a[10];
    printf("%d",*a+1-*a+3);
    return 0;
}
```

Output: 4

Explanation:
From operator precedence, de-reference operator has higher priority than addition/subtraction operator. So de-reference will be applied first. Here, a is an array which is not initialized. If we use a, then it will point to the first element of the array. Therefore *a will be the first element of the array. It’s effective value is 4.

What will be the output of the following C code?

```
#include <stdio.h>

int main(void)
{

    int b = 6;
    int c = 7;

    int a = ++b + c--;
    printf("%d", a);
    return 0;
}
```

What will be the output of the following C code?

```
#include <stdio.h>

int main(void)
{

    int b = 5 & 4 & 6;
    printf("%d", b);
}
```

a) 5
b) 6
c) 3
d) 4

```
#define prod(a,b) a*b
int main(void)
{
    int x=3,y=4;
    printf("%d",prod(x+2,y-1));
    return 0;
}
```

Output:

```
10
```

Explanation:
This program deals with macros, their side effects and operator precedence. Here prod is a macro which multiplies its two arguments a and b. Let us take a closer look.

prod(a, b) = a*b
prod(x+2, y-1) = x+2*y-1 = 3+2*4-1 = 3+8-1=10

If the programmer really wanted to multiply x+2 and y-1, he should have put parenthesis around a and b as follows.

prod(a,b) = (a)*(b)

This type of mistake in macro definition is called – macro side-effects.

```
int main(void)
{
    int i=0;
    while ( +(+i--) != 0)
        i-=i++;
    printf("%d",i);
    return 0;
}
```

Output:

```
-1
```

Explanation:

Let us first take the condition of while loop. There are several operators there. Unary + operator doesn’t do anything. So the simplified condition becomes (i–) != 0. So i will be compared with 0 and then decremented no matter whether condition is true or false. Since i is initialized to 0, the condition of while will be false at the first iteration itself but i will be decremented to -1. The body of while loop will not be executed. And printf will print -1.