Quiz Solution

Q 1

Which of the following are invalid variable names ? *

- ✗ ☐ BASICSALARY
- ✓ ✗ ☐ #MEAN — The only symbol can use is _
- ✗ ☐ group
- ✓ ✗ ☐ 422 — Can't be Number only Or began in number
- ✗ ☐ hELLO
- ✗ ☐ queue
- ✗ ☐ FLOAT — Because it's in capital letter so it can be used
- ✓ ✗ ☐ Plot # 3 — The only symbol can use is _
- ✗ ☐ _basic

Q 2

In this type of questions we should looking for:

1- Type of variables
2- The operations priorities

Evaluate the following expressions

1. g = big / 2 + big * 4 / big -big + abc/ 3 ; (abc= 2.5, big = 3, assume g to be a float)
2. on = ink * act / 2 + 3 / 2 * act + 2 + tig; (ink = 4, act = 1, tig= 3.2, assume on to be an int)
3. s = qui * add / 4 -6 / 2 + 2 / 3 * 6 / god ; (qui = 4, add = 2, god = 2, assume s to be an int)

g = **3 / 2** + 3 * 4 / 3 – 3 + 1.5 / 3 — operation: /
g = 1 + **3 * 4** / 3 – 3 + 1.5 / 3 — operation: *
g = 1 + **12 / 3** – 3 + 1.5 / 3 — operation: /
g = 1 + 4 – 3 + **1.5 / 3** — operation: /
g = **1 + 4** – 3 + 0.5 — operation: +
g = **5 – 3** + 0.5 — operation: -
g = **2 + 0.5** — operation: +
g = 2.5

on = **3 * 2** / 2 + 3 / 2 * 2 + 2 + 3.2
on = **6 / 2** + 3 / 2 * 2 + 2+ 3.2
on = 3 + **3 / 2** * 2 + 2+ 3.2
on = 3 + **1 * 2** + 2+ 3.2
on = **3 + 2** + 2+ 3.2
on = **5 + 2**+ 3.2
on = **7+ 3.2**
on = 10

s = **2 * 4** / 4 – 6 / 2 + 2 / 3 * 6 / 3 — operation: *
s = **8 / 4** - 6 / 2 + 2 / 3 * 6 / 3 — operation: /
s = 2 - **6 / 2** + 2 / 3 * 6 / 3 — operation: /
s = 2 - 3 + **2 / 3** * 6 / 3 — operation: *
s = 2 - 3 + **0 * 6** / 3 — operation: /
s = 2 - 3 + **0 / 3** — operation: -
s = **2 - 3** + 0 — operation: +
s = **-1 + 0**
s = -1

```c
/*
 * main.c
 *
 *  Created on: Dec 15, 201
 *      Author: keroles
 */

#include <stdio.h>
int main()
{
      float abc = 1.5 ;
      int big = 3 ;
      float g = big / 2 ;
        printf("%f", g);
        return 0;
}
```

```
1.000000
```

big = 3 int So when we / with int

The solution will be int = 1

abc = 1.5 float So when we / with int

The solution will be float = 1.5

```c
/*
 * main.c
 *
 *  Created on: Dec 15, 2018
 *      Author: keroles
 */

#include <stdio.h>
int main()
{
      float abc = 1.5 ;
      int big = 3 ;
      float g = big / 2.0 ;
        printf("%f", g);
        return 0;
}
```

```
1.500000
```

Q 3

```c
int i = 2, j = 3, k, l ;
float a, b ;
k = i / j * j ;
l = j / i * i ;
a = i / j * j ;
b = j / i * i ;
printf ( "%d %d %f %f", k, l, a, b ) ;
```

K int= 2/3*3 = int/int*int = 0*3=0

L int=3/2*2 =int/int*int = 1*2=2

A float=2/3*3= int/int*int =0*0=0.0

B float=3/2*2 = int/int*int =1*2=2.0

Print (0 2 0.0 2.0)

Q 4

programs are converted into machine language with the help of

(1) An interpreter
(2) A compiler
(3) An operating system
(4) None of the above

Compiler

Convert from language to binary all at once

Interpreter for ((web or desktop) applications (python))

Convert from language to another it's talking the operating system

Q 5

The real constant in C can be expressed in which of the following forms

(1) Fractional form only
(2) Exponential form only
(3) ASCII form only
(4) Both fractional and exponential forms

How C-language represent the real numbers (why? Because the all number in our live is fraction numbers not integer)

1- Fractional form (1.5)
2- Exponential (e) form (1-10e)

Q 6

In b = 6.6 / a + (2 * a + (3 * c) / a * d) / (2 / n) ; which operation will be performed first?

(1) 6.6 / a
(2) 2 * a
(3) 3 * c
(4) 2 / n

- () highest priority

Q 7

) Which of the following statement is wrong

(1) mes = 123.56 ;
(2) con = 'T' * 'A' ;
(3) this = 'T' * 20 ;
(4) 3 + a = b ;

- Not applicable and get Error

Q 8

Which of the following shows the correct hierarchy of arithmetic operations in C

(1) ( ), **, * or /, + or -
(2) ( ), **, *, /, +, -
(3) ( ), **, /, *, +, -
(4) ( ), / or *, - or +

- () highest priority

Q 9

If **a** is an integer variable, a = 5 / 2 ; will return a value

(1) 2.5
(2) 3
(3) 2
(4) 0

- A = int = int/int =2

Q 10

The expression, a = 7 / 22 * (3.14 + 2) * 3 / 5 ; evaluates to

(1) 8.28
(2) 6.28
(3) 3.14
(4) 0

- A =int/int*(int+int)*int/int
- (0)*(4.14)*0 = 0

Q 11

assume that a is int with 2 bytes *

- This is talking about the maximum Value can be add in this type of variable

The expression, a = 30 * 1000 + 2768 ; evaluates to

(1) 32768
(2) -32768
(3) 113040
(4) 0

- Int = 4 byte but here 2 bytes
- Signed int range 2 bytes
- The range of n bytes (-2^((8*n)-1)) to (2^((8*n)-1))-1)
- When n= 2 bytes = -32768 to 32767
- A = 32768 is out of max memory (out of the range) so we will take the expiration result then convert it in binary in rang of (2 bytes In Question) thin get the max of it by change it from +ve to -ve
- 32768 = 1000 0000 0000 0000 the +ve
- So the -Ve = 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1000 0000 0000 0000 but here there is only 2 bytes so we will take 1000 0000 0000 0000 only (16 bits)

- % above +
- 4 +(2) = 6

Q 12

e expression x = 4 + 2 % -8 evaluates to

(1) -6
(2) 6
(3) 4
(4) None of the above

- 5 % 3 = 2
- 5 % 3 = 2
- 5 % 3 = 2
- -ve /+ve = -ve
- -ve/-ve = +ve
- -ve%-ve = -ve
- +ve%-ve=+ve
- +ve/-ve = -ve
- The sign of the result of a remainder operation,

```
Problems  Tasks  Console  Properties
<terminated> (exit value: 0) EmbedXpro_C_1.exe [C/C++ App
2 -2 -2
-1 -1 1
```

```c
#include <stdio.h>
int main()
{
    float abc = 1.5 ;
    int big = 3 ;
    int g1 = 5 % (-3) ;
    int g2 = (-5) % (3);
    int g3 = (-5) % (-3);
    printf("%d %d %d \n", g1,g2,g3);
     g1 = 5 / (-3) ;
     g2 = (-5) / (3);
     g3 = (-5) / (-3);
      printf("%d %d %d", g1,g2,g3);
      return 0;
}
```

To calculate the remainder operation

1- We have a % b = a/b = int.float
2- Take the int then (a - (int * b)) = the remainder
3- 5 % 3 = 5/3 = 1.6 then the int = 1 so (5 – (1 * 3)) = 2
4- 2 % -8 = 2/-8 = 0.25 the int = 0 so (2 – (0 *-8)) = 2

Q 13

Assume that the size of char is 1 byte and negatives are stored in 2's complement form

```
#include<stdio.h>
int main()
{
    char c = 125;
    c = c+10;
    printf("%d", c);
    return 0;
}
```

- C = 125 + 10 = 135
- Char here = 1 byte = 8 bits = -2^((8*1)-1) = -128 to 127
- 135 out of rang so 135 = 1000 0111
- Get only 8 bits = 1000 0111 it's begin by 1 so it's -ve number so then covert it 0111 1001 = 121 the add -ve = -121
- Will print -(-121) because char signed data type range (+/-)

Another solution for signed data types only

- C = 125 + 10 = 135
- Char here = 1 byte = 8 bits = -2^((8*1)-1) = -128 to 127
- So 128 + 127 = 255 +1 (because there is a 0 number we should add it) = 256

Q 14

Predict the output of following program. Assume that the numbers are stored in 2's complement form. *

```
#include<stdio.h>
int  main()
{
   unsigned int x = -1;
   int y = ~0;
   if (x == y)
      printf("same");
   else
      printf("not same");
   return 0;
}
```

- X = Unsigned (+ve) only
- We will assume the int is = 2 bytes
- 1 = 0000 0000 0000 0001 so -1 = 1111 1111 1111 1111
- Every 1111 = f so it will be stored in memory = 0xffff
- Y = ~0 = not 0 = 1111 1111 1111 1111 = 0xffff
- So when say y == x that is mean it will go to the memory and comparison the 2 address y = 0xffff and x = 0xffff so will print same

Q 15

Predict the output *

```
#include <stdio.h>

int main()
{
   float c = 5.0;
   printf ("Temperature in Fahrenheit is %.2f", (9/5)*c + 32);
   return 0;
}
```

- C is float = 5.0
- 9/5 = int / int = 1
- 1*5.0 = 5.0 + 32 = float + int = 37.00

(A) Temperature in Fahrenheit is 41.00

(B) Temperature in Fahrenheit is 37.00

(C) Temperature in Fahrenheit is 0.00

(D) Compiler Error

Q 16

```
#include <stdio.h>
int main()
{
    char a = '\012';

    printf("%d", a);

    return 0;
}
```

- C is char = 'a' = 1 byte = 97 (from ASCII)
- Char = '\012' that is main get the character that equivalent number 12 (as Octa (OCT)) in Decimal
- So 12 in decimal = 10 in ASCII code 10 = B
- We need to print %d (int) so will print 10
- That is main when we see \0(number) that is main get the octa number (number) and get the decimal of it and convert is using ASCII to character

(A) Compiler Error

(B) 12

(C) 10

(D) Empty

Q 17

Predict the output of the below program: *

```c
#include <stdio.h>
int main()
{
    printf("%d", 1 << 2 + 3 << 4);
    return 0;
}
```

- Bit shifting is not multiplication. It can be used in certain circumstances to have the same effect as a multiplication by a power of two
- 1<<5<<4 == 1<<9 == 1*2^9 =512

Explanation:

- The main logic behind the program is the precedence and associativity of the operators. The addition(+) operator has higher precedence than shift(<<) operator. So, the expression boils down to 1 << (2 + 3) << 4 which in turn reduces to (1 << 5) << 4 as the shift operator has left-to-right associativity.

```c
/*
 * main.c
 *
 *  Created on: Dec 15, 2018
 *      Author: keroles
 */

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

    printf("%d \n", 1 + 2 * 8);
    printf("%d \n", 1 + 2 << 3);


    return 0;
}

```

```
17
24
```

- + higher that <<
- So 1+2*8=17 but 1+2<<3=3<<3=3*(2)^3=24
- A << b main = a * (2)^b and A >> b main a/(2)^b
- So 1 *(2)^(5)=32*2^4=512
- When we se << and << we can add them to first number
- 1*2^(5+4)=512

Q 18

```c
#include <stdio.h>

int main()
{
    int i = (1, 2, 3);

    printf("%d", i);

    return 0;
}
```

- If we don't put () that is mean I = 1 then I = 2 then I = 3 so the final definition of I is 3

Q 19

```c
#include <stdio.h>
int main()
{
    int i = 5, j = 10, k = 15;
    printf("%d ", sizeof(k /= i + j));
    printf("%d", k);
    return 0;
}
```

- Sizeof equation looking only for the type of data in the () so if there is any operations in the () as like this question
- We should know the type of data out from this operation
- Here the type of the data out is int
- So the size of int = 4 byte
- Then k = 15 because the variable k the same because of sizeof equation so the result = 4 15

Q 20

Notes

- Int a,b,c then A = b =c = 100 that is mean the all variables are declared but
- Int a = b = c = 100 that is mean a only declared but b and c not declared
- So a == b mean 100 == 100 that is true = 1 then
- 1 == c that is mean 1 == 100 no so it's false

- See the program, the values of a,b and c is 100 and you are thinking how condition is false here and why output is "False..."?
- The expression isa==b==c which will evaluates like(a==b)==c now what will be the result?
- •The result of(a==b)is1(i.e. true).
- •And(1)==c will be0(i.e. false) because the value ofcis 100 and100 is equal not to 1.

main.c

```c
/*
 * main.c
 *
 *  Created on: Dec 15, 2018
 *      Author: keroles
 */

#include <stdio.h>
int main(){
    int a,b,c;
    a=b=c=100;

    if(a==b==c)
        printf("True...\n");
    else
        printf("False...\n");

    return 0;
}
```

Problems Tasks Console Properties AVR Device Explorer AVR Supported

<terminated> (exit value: 0) EmbedXpro_C_1.exe [C/C++ Application] C:\Users\kkhalil\avr_ws\EmbedXpr

```
False...
```

```c
#include <stdio.h>
int main()
{
  int a = 10, b = 20, c = 30;
  if (c > b > a)
    printf("TRUE");
  else
    printf("FALSE");
  return 0;
}
```

- a==b==c (Multiple Comparison) evaluates in C programming
- (==) operates from left to right
- Expression a==b==c is actually (a==b) ==c
- There int a =10 , b =20 , c = 30
- C > b = 30 > 20 ---- True and true = 0 then true > a = 1 > 10
- ----- False

Q 21

```c
#include <stdio.h>
int main()
{
   int a = 0;
   int b;
   a = (a == (a == 1));
   printf("%d", a);
   return 0;
}
```

- A = 0 and b declared only
- A == 1 ---- false = 0
- False (0) == a ---- True = 1
- So a = 1

Q 22

- A &= ~(1 << bit) — Reset
- (A & 1 << bit) != 0

Q 23

- A |= 1 << bit — set
- A & ~B

Q 24

- A ^= 1<<bit — Toggle

Implicit / Explicit concept

.....Conversion is done programatically. *

- Implicit
- Explicit
- other

الطريقة البرمجية التي تقوم بعملها -

type casting is to *

- Implicit
- Explicit
- other

الذي تقوم بتعريف فيه المتغير -

.....Conversion is done automatically.

- Implicit
- Explicit
- other

لا دخل لك فيه -

Conversion of larger number to smaller number is ..... conversion.float k=123.456int i= (int) k

- Implicit
- Explicit
- other

Conversion of smaller number to larger number is .....conversion.Conversion of integer type data to float.float i=0;int j=10;i=j;

- Implicit
- Explicit
- other

Q 30

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

- 3 6
- compiler Dependent
- 3 4
- 3 3

- The reason for undefined behaviour in PROGRAM 1 is, the operator '+' doesn't have standard defined order of evaluation for its operands
- Explanation: See : https://www.geeksforgeeks.org/sequence-points-in-c-set-1/ for explanation.

- A = 1 post a++ = 2
- Now a = 2 then a++ = 3
- So b should be = 3
- But here we assume that the compiler do the left operation before the right one
- But the right solution not clear in this point because it's depended on compiler
- If there is no choice we will assume our solution from left to right but in printf condition (from right to left)

Q 31

```c
#include <stdio.h>
int main()
{
   int x = 10;
   int y = (x++, x++, x++);
   printf("%d %d\n", x, y);
   return 0;
}
```

- 13 12
- compiler Dependent
- 13 13
- 10 10

- If Y =(number , number , number)----- we take the last number because () mean you tell the compiler to get step by step y = (1,2,3)--- mean
- Y =1 then y = 2 then y =3 ---- step by step
- So here y = (11 , 12 ,13) ---- y = 12 in printf and x =13 because x++ postfix so we will print y =12

Q 32

```c
#include <stdio.h>
int main()
{
    int i = 3;
    printf("%d", (++i)++);
    return 0;
}
```

- 3
- 4
- 5
- compile error

- Explanation: In C, prefix and postfix operators need l-value to perform operation and return
- r-value. The expression (++i)++ when executed increments the value of variable I (I is a l-value) and returns r-value. The compiler generates the error (l-value required) when it tries to post-increment the value of a r-value.

- Compiler error because ++I that it's mean I will get I = 3 then I will add it I = 4 then I will take 4 to store it in memory so it will be (4)++ --- 4 = 4 +1 --- here we can't store number in 4

Q 33

Predict the output of the following code ?

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

- In printf go from right to left ---- I++ = 11 but because it's postfix it will print 10
- And at the prefix it will be return 12
- 12 10

- 12 10
- 12 12
- 12 11

- In case of GCC Compiler Output will be 12,10. Output may very from compiler to compiler because order of evaluation inside printf

- Output may Vary from Compiler to Compiler

Q 34

Optimize

int X,i=4,j=7; X=j || i++ && 1

- X=1 , i =4
- X=7 , i =4
- X=7 , i =5
- compile error

- X = (j || (i++ && 1)); Note that precedence does not equate to order of execution in general. In this case, we have the following evaluation logic:
- To evaluate = we need to evaluate its right-hand upper and To evaluate (j || stuff...) we first evaluate j j is non-zero, so the result of (j || stuff...) is 1, and we do not evaluate stuff due to the short-circuit behaviour of ||Now we have determined that the right-hand operand of = has evaluated to 1, so assign 1 to X.Final result: X == 1, and I and j unchanged.

- When we see X (true value mean = 1) || any value that is mean it will = 1 that is call optimized x =1
- i= 4 because we clear all of this so nothing happened in i
- optimized mean the compiler execute the time and delete this item

Q 35

```
void main()
{
        int i=0, j=1, k=2, m;
        m = i++ || j++ || k++;
        printf("%d %d %d %d", m, i, j, k);
}
```

- I = 0 so we will return I first = 0 because of postfix but in memory I = 1 so
- M = 0 || 1 || 2 --- 1 || K++ any True || with anything will get True so will = 1 always
- 1 1 2 2 ---- k = 2 because this term optimized

- 1123
- 1122
- 0122
- 0123
- None of these

Answer: Option B Solution:

- In an expression involving || operator, evaluation takes place from left to right and will be stopped if one of its components evaluates to true (a non-zero value).
- So, in the given expression m = i++ || j++ || k++. It will be stop at j and assign the current value of j in m.

Q 36

. What will be the output of the following code fragment?

```
void main()
{
   printf("%x",-1<<4);
}
```

- fff0 ✓
- fff1
- fff2
- fff3
- fff4

Answer: Option A Solution:

- 1 will be represented in binary form as:1111 1111 1111 1111
- Now -1<<4 means 1 is Shifted towards left by 4 positions, hence it becomes:1111 1111 1111 0000 in hexadecimal form -fff0

- **Write “Hello World” without semicolon**

- If (printf (“Hello World\n”)) or If (printf (“Hello World\n”)){}
- When we print anything (“Hello World\n”) that is mean it will return 10 according to number of characters
- And any value = true so the condition = true

- **Error in execution is**

- Runtime Error
- (syntax error) Source code --- compiler (ex found that for equation wrote wrong)
- (logical error) Source code --- compiler ----binary then it will burn on flash memory inside the microcontroller then it will be run but do something wrong
- (linker error) Source code --- compiler ----Linker (didn’t found the equation that you called in your source code and that happened before get the binary file
- (semantic error) Source code --- compiler ---- binary (ex define function char but when you call it you call it as int so (char = 1 byte and int = 4 bytes)
- (runtime error) Source code --- compiler ----Linker and you get your binary file then put it on the microcontroller then you run it (but in your code you defined something (pointer) you don’t have any access at it or out of the memory) so crash at runtime will happened

- **What**’s the output:
- **#include <stdio.h>**
- **Void main ()**
- **{**
- Int I = ox10+010+10;

- If you want to write an octa number out before it 0 so
- 010 = octa number
- 0x10 = decimal number
- 10 = hexadecimal number
- 0x10 = 10 ----- decimal
- 010 = 8 ----- decimal
- 10 = 16 ----- decimal
- We will get 34 ----- decimal so = 22 ----- hexadecimal

- **___type___var___=___value___;**

- Declaration : int x; (variable with type) (ex when we write extern int x;" that's mean we till the compiler there is an variable named x with type int but in anther file so let the linker link this file to defined this variable so now there is no memory saved to this variable till the linker link the other file to f=define it
- then, got to the memory and save an area at it = 4 bytes with corrupted data (definition)
- (Initialization) when say x = 4; that's mean go to at x and put it by 4 not 0

- **Extern__type___var___;**

- Declaration

- **A = 5 , b = 4 if(a==b); printf (" Equal ");**

- Print "Equal "
- Because there is ; after if (); so that will = null (it will be ignored) so it will go to the next code line

- **Unsigned short I =0xffff; while (i++ != 0) printf("%d", ++i);**

- Short can store 0xffff = 2 bytes
- When i++ = 0xffff + 1 overflow will happened so it will return to 0
- ++ 0 = 1 so you get over 0 and you will not get it
- So it's infinite loop

- **Int x=3; float y=3.0; if (x==y){printf("True");}**

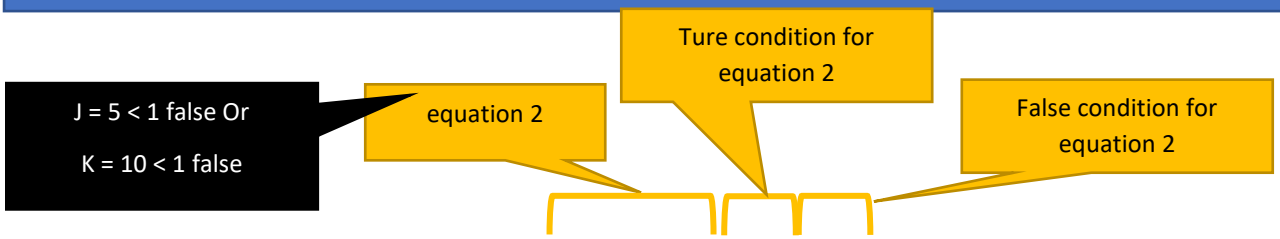
- **Printf "True"**

- **Int i=1; if (i++&&(i==1))**

- I = 1 then first (1==1) true
- 1++ = 2 && 1
- ++ higher that ()
- false

- **Char ch =’a ’=97 ; switch (ch) {case 97 ; Printf(“97”) ; break; case ‘a ‘; printf (“a “) ; break;}**

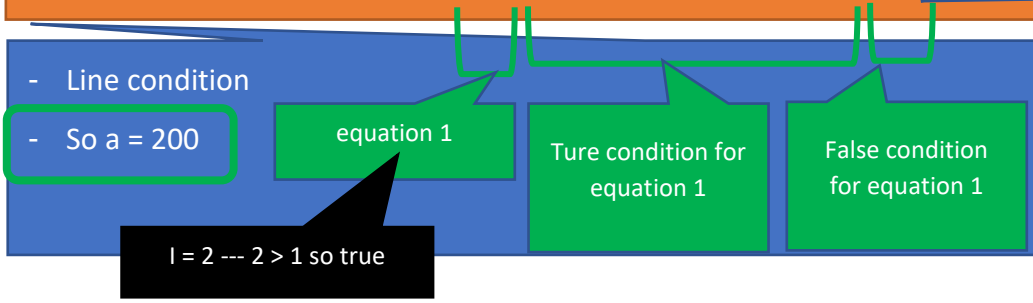
- Switch (x) x cant be float
- and can’t do case x and case x
- here same cases with the same number so it will be compiler error



- **Int I = 2 , j = 5 ,k = 10 ; a = I > 1 ? J < 1 || k < 1 ? 100 : 200 : 300 ;**

The last condition always the false one

- Line condition
- So a = 200



- Write Source Code to Swap Two Numbers without temp variable.

- X= 5 , y =2
- X = x +y-----7
- Y = x – y-------5
- X =x – y--------2
- Finally x =2 , y= 5