****

**C/C++ TEST**

**Time allowed: 30 minutes**

**NAME:**

**DATE:**

**Question # 1**

In C++, what gets printed?

void print(int x = 1, int y = 2 , int z = 3)

{

cout << x << y << z;

}

int main()

{

print(), print(4), print(5, 6) , print(7, 8, 9);

return( EXIT\_SUCCESS);

}

1. 123
2. 456789
3. 123456789
4. It won't compile.
5. 123423563789

**Question # 2**

Predict the output from:

if (5 < 4)

if (6 > 5)

putchar('1');

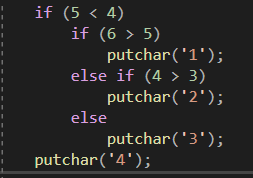
else if (4 > 3)

putchar('2');

else

putchar('3');

putchar('4');



1. 4
2. 2
3. 24
4. 4 or 24, depending upon the implementation
5. Nothing is printed.

**Question # 3**

In C which statement is true concerning a mojor problem with the following?

long double fx(void)

{

double answer = sum(1.1, 2.2 , 3.3);

return printf("answer = %f", answer);

}

double sum(double a, double b, double c)

{

return(a + b + c);

}

// C dịch từ trên xuống, khi dịch tới answer = sum sẽ ko tìm thấy hàm sum ở dưới => ko tìm thấy và gọi đến sum của ANSI tiêu chuẩn

1. The name sum conficts with a standard ANSI math function.
2. The return statement in fx return type double.
3. Return statements may not contain an algebraic expression (a + b + c).
4. There is nothing wrong with the program.
5. The call to sum assumes that sum returns type int.

**Question # 4**

In C++, what gets printed?

void print(int x = 1, int y = 2 , int z = 3) { cout << x << y << z; }

void print(long x, int y = 5 , int z = 6) { cout << x << y << z; }

int main()

{

print(4), print(4L) ;

return( EXIT\_SUCCESS);

}

1. 44L
2. 423423
3. 423456
4. Output is implementation(thực hiện) dependent.
5. It won't compile because the print function definition are ambigous(mơ hồ).

**Question # 5**

If a prototype(mẫu) for fx (below) is present, predict(dự đoán) the output from:

printf("%d", \*fx)

int \*fx(void)

{

int x = 5;

return(&x);

}

1. 5
2. garbage
3. the address of the variable x
4. A compile error occurs
5. none of the above or implementation dependent.

**Question # 6**

What is the main problem with the following:

int ip[]= {6,7,2,4,-5};

for (int i=0; i<5; ++I; ++ip)

cout << \*ip; //gtri ptu ip[0]

//ip là địa chỉ của của mảng ip[] => nó là hằng

1. Nothing is wrong
2. An unitialized(đơn vị) pointer is being dereferenced(hủy đki)
3. An attempt(nỗ lực) is being made to modify(sửa) the name of an array, a constant
4. It contains a magic number, which is illegal in some compilers

**Question # 7**

What will be the result when you attempt to compile and execute this program?

#include <stdio.h>

void main()

{

printf("%d", 2["abcd");

}

1. Compile time error
2. garbage
3. 99
4. 2
5. none of the above

**Question # 8**

What will happen when you attempt to compile and run the following code:

#include <stdio.h>

#define MAX 20;

void main()

{

int a[MAX] = {1,2};

printf("%d", a[1]);

}

//define ko có ; đằng sau

1. 1
2. 2
3. 0
4. It won't compile
5. none of the above

**Question # 9**

What will happen when you attempt to compile and run the following code:

#include <stdio.h>

void swap(int \*p1, int \*p2)

{

\*p1 ^= \*p2; //XOR bit

\*p2^= \*p1;

\*p1^= \*p2;

}

void main()

{

int a = 5, b = 6, c = 7;

swap(&a, &b);

swap(&c, &c);

printf("%d, %d, %d", a, b, c);

}

1. 5,6,7
2. 6,5,7
3. 5,6,0
4. 6,5,0
5. none of the above

**Question # 10**

What will happen when you attempt to compile and run the following code:

#include <stdio.h>

int sum(int \*\*a, int m, int m)

{

int i, j, s = 0;

for (i = 0, i < m; i++)

for (j = 0; j < n; j ++)

s+=a[i][j];

return s;

}

void main()

{

int a[2][2] = {{1,2}, {3,4}};

int m = 2, n =3;

printf("%d", sum(a,m,n));

}

// mảng chỉ phân rã 1 lần thành con trỏ => ko ép được a[][] thành \*\*a để vào hàm, có thể chuyển:

void display(int (\*p)[numcols],int numRows,int numCols)//First method//

void display(int \*p,int numRows,int numCols) //Second Method//

void display(int numRows,int numCols,int p[][numCols]) //Third Method

1. 10 is printed out
2. 0 is printed out
3. nothing is printed out
4. It won't compile
5. none of the above

**Question # 11 //Con trỏ hàm:**

<return\_**type**> (\*<name\_of\_pointer>)( <data\_type\_of\_parameters> );

How to declare(khai báo) an array of 3 pointers to functions to functions returning int

//mảng gồm 3 con trỏ. Mỗi con trỏ, trỏ tới 1 hàm trả về kiểu int

Int ham(…){

Return 3;

}

1. int\*\*a(3);
2. int(\*a)(3);
3. int(\*a[3])();
4. int(\*a)[3];
5. none of the above

**Question # 12**

What will happen when you attempt to compile and run the following code:

#include <stdio.h>

void print()

{

#ifdef \_D

printf("Debug");

#else

printf("Release");

#endif

}

#define \_D

void main()

{

print();

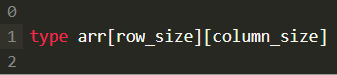
}

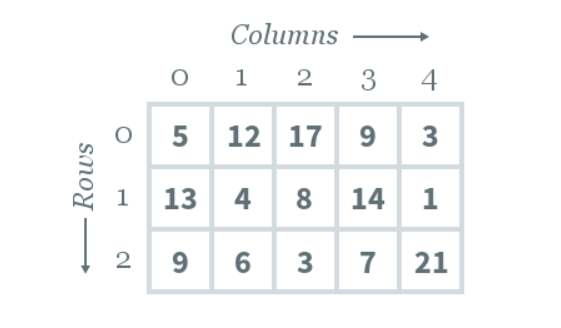
//#ifdef \_D: nếu \_D đã đc #define \_D ở trước hàm print()

1. Debug is printed out
2. Release is printed out
3. nothing is printed out
4. It won't compile
5. none of the above

**Question # 13**

Declare a multi dimensioned array of floats called balances having three rows and five columns:





1. float balances[3][5]
2. balances[3][5] of float
3. float balances [5][3]
4. array of float balances [0….2][0…5]
5. float balances [3,5]

**Question # 14**

Assuming a 16 bit int 2's complement implementation, presict the value of: -17 >> 1

// 17 = 16+1= 0000 0000 0001 0001

//-17 = = 1111 1111 1110 1110

//-17 >>1 = 1111 1111 1111 0111 = -9 (-17/2)

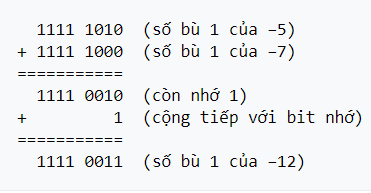
//-17 <<1 = 1111 1111 1101 1110 = -34 (-17x2)

1. -9 or 0x7FF7
2. -8
3. 17
4. 8
5. other implementation dependent values

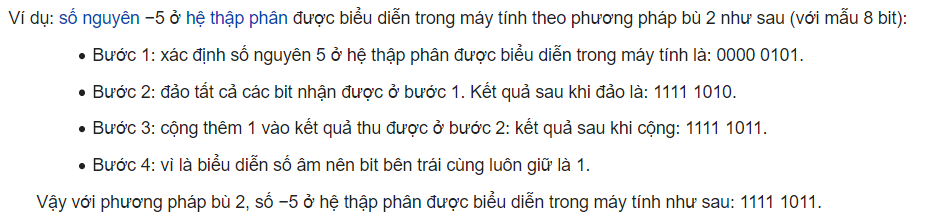
**// Cần đọc lại phần bù 1 và phần bù 2:**

* **Bù 1:**

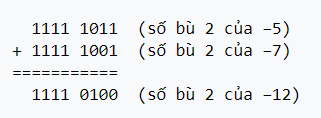
**7 = 0000 0111 => -7 =1111 1000 (đảo bit)**



* **Bù 2:** = Bù 1 + 1



* **Khi cộng có bit tràn – bỏ qua**



**Question # 15**

If an int is 16 bits and a char is 8 bits, the values is sch and uch after signed char sch = 256; and unsigned char uch = 256; are:

//char -128->127

//unsigned 0-255

//char sch = 256 = -128 + 256 -127 -1=0

//uch = 0 +256 -255 -1=0

1. sch is 256 and uch is 256
2. sch is implementation defined and uch is 256
3. sch is implementation defined and uch is 0
4. sch is 0 anf uch is 0
5. the results of both are undefined

**Question # 16**

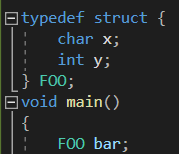
On a machine using 1's complement negative integaers and 16 bit ints, what is the bit pattern for -2?

2= 0000 0000 0000 0010 => -2 (bù1)= 1111 1111 1111 1101

1. 1111 1111 1111 1111
2. 1111 1111 1111 1110
3. 1111 1111 1111 1101
4. 1000 0000 0000 0010
5. implementation dependent

**Question # 17**

For typedef struct {char x; int y;} FOO;FOO bar; which of the following may be false?



1. sizeof(FOO) == sizeof(bar)
2. sizeof(FOO) == sizeof(x) : 1 + sizeof(y): 4 ( 8 == 5 )
3. &bar is numerically equal to &bar.x
4. (char\*)&bar + offsetof(FOO,y) == (char\*)&y
5. they can all be false, depending upon(=on) implemention

**Question # 18**

What is wrong with the following string initialization? Char s[] = {'H', 'E', 'L', 'L', 'O', NULL};

1. Nothing is wrong
2. The syntax(cú pháp) is incorrect
3. A character array can't hold a string
4. NULL may be of the wrong type and its value is not necessarily even equal to 0.
5. Strings can't be initialized(khởi tạo)

**Question # 19**

Assuming(giả sử) #define sum(a, b) a + b predict the value of: 5 \* sum( 3 + 1, 2)

//5 \* sum( 3 + 1, 2) = 5\*3 +1 +2 = 18

1. 30
2. 18
3. 22
4. none of the above
5. implementation dependent.

**Question # 20**

What is the main problem with the following: int \*ip; for ( \*ip = 0; \*ip < 5; \*ip++) ; //lỗi thực thi



1. Nothing is wrong.
2. It dereferences(hủy bỏ) an uninitialized(chưa khởi tạo) pointer.
3. It does nothing useful.
4. Int contains a magic number.
5. It contains implementation(thực thi) dependent problem(s).

**Question # 21**

In C with no prototype , what data types get passed to fcn by the call: fcn((char)23, (short)34, 87, 6.8f)

1. char, short, int, float
2. char, short, long, float
3. int, int, int, float
4. int, int, int, double
5. none of the above or implementation dependent.

**Question # 22**

Predict what gets printed by: cout << (12 < 5 ? "Hello " : "World")

1. Hello
2. Hello World
3. World
4. World Hello
5. Output is undefined or implementation dependent.

**Question # 23**

Predict final value of i: for ( int i = 0; i < 5 ; i++) break;

1. 0
2. 1
3. 2
4. 3
5. none of the above.

**Question # 24**

Predict what gets printed by: printf("Goodbye") && printf("Cruel") || printf("World")

1. Goodbye
2. GoodbyeCruel
3. Goodbye Cruel World
4. Goodbye World
5. Output is implementation dependent.

**Question # 25**

Predict what gets printed: const int I; for (int = 0; i < 5; ++i) cout << i << ' ' ;

1. 0 1 2 3 4
2. 0 1 2 3 4 5
3. 1 2 3 4 5
4. It won't compile
5. Output is implementation dependent.

**Question # 26**

The values of -5/4 and -5%4, respectively, are:

1. implemntation dependent: -5/4 == -1 and -5%4 == -1 or -5/4 == -2 and -5%4 == 3
2. -1 and -1
3. -2 and 3
4. -1 and -2
5. none of the above.

**Question # 27**

Predict the output from cout << oct(hệ8) << 15 << dec(hệ 10) << 15 << hex(hệ16) << 15:

1. oct 15 dec 15 hex 15
2. 017 15 0xf
3. 17 15 f
4. 1715f
5. Output is implementation dependent.

**Question # 28**

Assuming a 1G bit type int and a 32 bit type long, the data types of 32767, -32678,32768 and 2. are:

1. int, int , long ,float
2. int, long, long, float
3. int, long, long, double,
4. implementation dependent
5. none of the above.

**Question # 29**

The value of sizeof('A') is always:

1. The same as the value of sizeof(char)
2. The same as sizeof(int) in C and the same as sizeof(char) in C++
3. 65 if the ASCII character set is used.
4. Dependent upon the character set being used.
5. None of the above.

**Question # 30**

In C, if variables x,y and z are properly(đúng) declared(khai báo), what is syntactically(cú pháp) wrong with:

z = y//\* division\*/ x;

1. Nothing is wrong.
2. Everything after the // is a comment so the statement is incomplete.
3. It is not portable
4. A comment may not serve as whitespace.
5. The value of y may be too large.