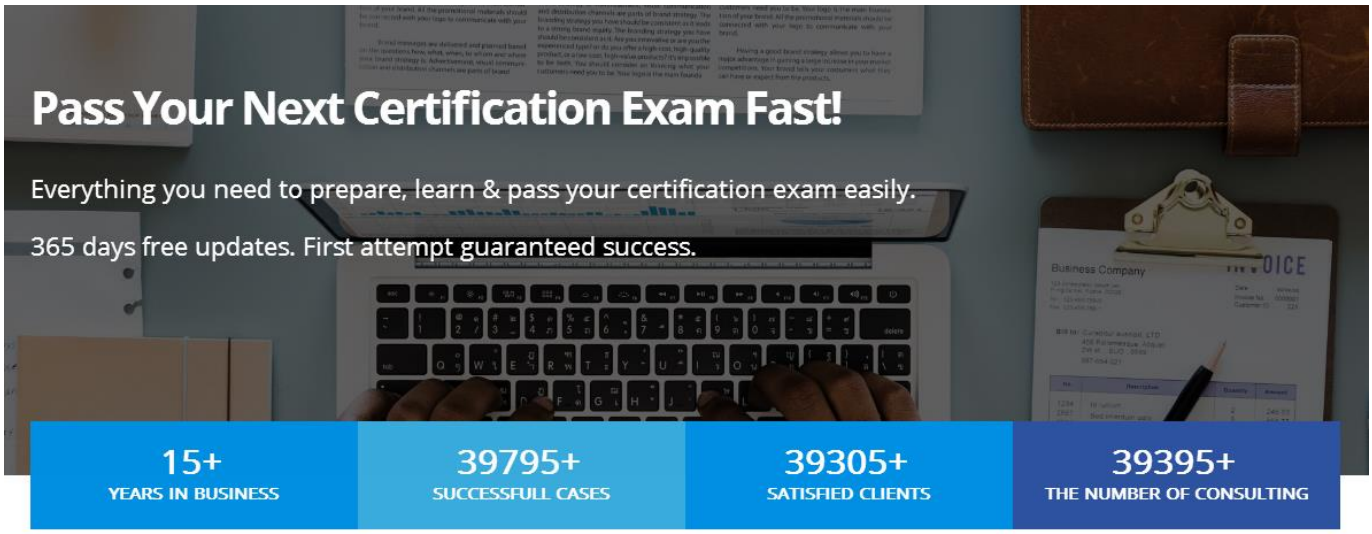
PracticeVCE



TRY BEFORE YOU BUY

Download a free sample of any of our exam questions and answers

- 24/7 customer support, Secure shopping site
- Free One year updates to match real exam scenarios
- If you failed your exam after buying our products we will refund the full amount back to you.

365 Days Free Updates

Free update is available within 365 days after your purchase. After 365 days, you will get 50% discounts for updating.

Security & Privacy

We respect customer privacy. We use McAfee's security service to provide you with utmost security for your personal information & peace of mind.

Instant Download

After Payment, our system will send you the products you purchase in mailbox in a minute after payment. If not received within 2 hours, please contact us.

Money Back Guarantee

Full refund if you fail the corresponding exam in 60 days after purchasing. And Free get any another product.

http://www.practicevce.com

Professional Study Tool and Reliable Exam Practice Material

Exam : **CLA-11-03**

Title : CLA - C Certified Associate Programmer

Vendor : C++ Institute

Version : DEMO

NO.1 What happens if you try to compile and run this program?

```
#include <stdio.h>
int main (int argc, char *argv[]) {
int i =2, j = 1;
if(i / j)
j += j;
else
i += i;
printf("%d",i + j);
return 0;
}
```

Choose the right answer:

A. The program outputs 4

B. The program outputs 3

C. Compilation fails

D. The program outputs 1

E. The program outputs 5

Answer: A

Explanation:

In the if statement, i / j is 2 / 1, which is true. Therefore, the if block is executed, and j += j; doubles the value of j (j becomes 2).

After the if-else statement, printf("%d", i + j); prints the sum of i and the updated val-ue of j (2 + 2), which is

4.

NO.2 What happens when you compile and run the following program?

```
#include <stdio.h>
#define SYM
#define BOL 100
#undef SYM
int main (void) {
#ifdef SYM
int i = 100;
#else
int i= 200;
#endif
int j = i + 200;
printf("%d",i+j);
return 0;
}
```

Select the correct answer:

A. The program outputs 200

B. The program outputs 600

C. The program outputs 100

D. The program outputs 400

E. The program outputs 300

Answer: B

Explanation:

The program outputs 600 because the #ifdef directive checks if the macro SYM is defined, and if so, executes the code between it and the corresponding #else or #endif directive. Otherwise, it skips that code and executes the code after the #else directive, if any. In this program, the macro SYM is defined by the #define directive, but then undefined by the #undef directive, which removes the definition of a macro. Therefore, the code between the #ifdef and the #else directives is skipped, and the code after the #else directive is executed, which assigns 200 to the variable i. The variable j is then assigned the sum of i and 200, which is 400. The printf function then prints the sum of i and j, which is 600, as a decimal integer using the %d format specifier.

References = CLA - C Certified Associate Programmer Certification, C Essentials 2 - (Intermediate), C Preprocessor

NO.3 What happens if you try to compile and run this program?

```
#define ALPHA 0
#define BETA ALPHA-1
#define GAMMA 1
#define dELTA ALPHA-BETA-GAMMA
#include <stdio.h>
int main(int argc, char *argv[]) {
printf ("%d", DELTA);
return 0;
```

Choose the right answer:

A. The program outputs -1

B. The program outputs 1

C. The program outputs 2

D. Compilation fails

E. The program outputs -2

Answer: D

Explanation:

Let's analyze the macros and the program:

1.ALPHA is defined as 0.

2.BETA is defined as ALPHA - 1, which is 0 - 1.

3.GAMMA is defined as 1.

4.DELTA is defined as ALPHA - BETA - GAMMA. With the previous definitions, this expands to 0 - (0 - 1) -

1.

Now, let's expand DELTA with the given values:

makefileCopy code

DELTA = 0 - (0 - 1) - 1 DELTA = 0 - 0 + 1 - 1 DELTA = 0 + 1 - 1 DELTA = 1 - 1 DELTA = 0 It is important t o note that the macro dELTA is defined with a lowercase 'd', but the printf function is trying to print DELTA with an uppercase 'D'. Preprocessor tokens are case-sensitive, so this is a mismatch. However, for the sake of the question, let's assume that dELTA was meant to be DELTA with an uppercase 'D'. Since the actual calculation results in 0, but there is a typo in the printf statement (it should print

dELTA, not DELTA), the compilation will fail due to DELTA not being defined.

NO.4 What happens if you try to compile and run this program?

```
#include <stdio.h>
int main (int argc, char *argv[]) {
int i = 2;
int d= i << 2;
d /= 2;
printf ("%d", d) ;
return 0;
}
```

Choose the right answer:

A. The program outputs 1

B. Compilation fails

C. The program outputs 0

D. The program outputs 2

E. The program outputs 4

Answer: E

Explanation:

The program outputs 4 because the expression i << 2 performs a left shift operation on the binary representation of i, which is 00000010, by two bits, resulting in 00001000, which is equivalent to 8 in decimal.

Then, the expression d /= 2 performs a division assignment operation, which divides d by 2 and assigns the result back to d, resulting in 4. The printf function then prints the value of d as a decimal integer using the %d format specifier.

References = CLA - C Certified Associate Programmer Certification, [C Essentials 2 - (Intermediate)], C Bitwise Operators, C Assignment Operators

NO.5 What happens if you try to compile and run this program?

```
#include <stdio.h>
int main (int argc, char *argv[]) {
int i = 20;
printf("%x", i);
return 0;
}
```

-

Choose the right answer:

A. Compilation fails

B. The program outputs 14

C. The program outputs 10

D. The program outputs 24

E. The program outputs 20

Answer: B

Explanation:

The program outputs 14 because the printf function prints the value of i as a hexadecimal integer

using the %x format specifier. The hexadecimal system uses 16 symbols to represent numbers, from 0 to 9 and from A to F.
Each symbol corresponds to a decimal value, for example, A is 10, B is 11, C is 12, and so on. To convert a decimal number to a hexadecimal number, we need to divide the number by 16 repeatedly and write down the remainder in reverse order. For example, to convert 20 to hexa-decimal, we do:
20 / 16 = 1, remainder 4 1 / 16 = 0, remainder 1
The hexadecimal number is 14, as we write the remainders from right to left.
References = CLA - C Certified Associate Programmer Certification, C Essentials 2 - (Intermediate), C printf and scanf functions, Hexadecimal number system