Question 1 of 20
The program:

```
#include <stdio.h>
void main() {
        for(int i = 1; i< 4; i++) {
                static int j = 1; int k = 1;
                printf("\n i-%d j-%d k-%d", i, j, k);
                j++; k++;
        }
}
```

displays:

A. 1 1 1
   2 2 1
   3 3 1
B. 1 1 1
   2 1 1
   3 1 1
C. 1 1 1
   2 2 2
   3 3 3
D. 1 1 1
   2 1 2
   3 1 3
E. Other combination of values

Question 2 of 20
Which of the following elements isn't part of the "strong name" concept:

A. Version number
B. Culture with which it is associated
C. Public key
D. Digital signature
E. Security policy

Question 3 of 20
Specify what displays the program below:

```
#include <stdio.h>
void main() {
   int *aux = new int, i, **tabela = new int*[3];
   for(i = 0; i < 3; i++) {
      *aux = i;
      tabela[i] = aux;
   }
   for(i = 0; i < 3; i++) {
      printf("%d", *tabela[i]);
   }
```

}

A. 1 2 3
B. 0 1 2
C. 2 2 2
D. 1 1 1
E. 3 3 3

Question 4 of 20

```
#include <stdio.h>
void main() {
    int x = 7, y = 8, &ri = x;
   printf("\n %d ", ri);
   ri = y;
   y = 10;
   printf("\n %d ", ri);
}
```

The sequence above:

A. Is wrong, assignment ri = y; is not permitted;
B. Displays 7 10, because ri is referring now y;
C. Displays 7 8, because a reference can't be changed, only be referred as variable;
D. Is wrong, a reference can't be displayed;
E. Displays 10 10, the last reference initialization is that retained by the compiler

Question 5 of 20

An attack that only allows the attacker to inhibit a service, without gaining anything else, is known as:

A. Denial of Service (DoS)
B. Service Linking
C. Watermarking
D. Obfuscation
E. Machine "owning"

Question 6 of 20

What is the best way to secure and reduce as size the programs, based on dynamic link libraries programs (DLLs)?

A. RSA Encryption
B. Assembly Linking
C. Watermarking
D. Library Bypass Obfuscation
E. Fingerprinting

Question 7 of 20

The program becomes easier decompiled, because Visual C#.NET is compiled firstly into an intermediate language?

A. No, because the intermediate language is more difficult to understand by an attacker
B. Yes, because restoring the programming structures is much easier
C. No, because the intermediate language is platform dependent running
D. Yes, because the intermediate language allows running on a virtual machine

E. C#.NET language is scripting type and is not compiled, but is interpreted

Question 8 of 20
Obfuscators alter the source code and make safer programs, but cannot be debugged.

A. Yes, because it cannot restore the original context of the program
B. No, because obfuscators change the source code, but keep some original information too;
C. Yes, because searching in tables of association is no longer possible
D. Obfuscated programs cannot be repaired
E. Obfuscation doesn't make safer programs

Question 9 of 20

```
#include <iostream.h>
class dst {
public:
   int x;
};
class srs {
private:
   int x;
public:
   srs(int a): x(a){}
   operator dst() {
      dst d;
      d.x = x;
      return d;
   }
};
void main() {
   srs s(1);
   dst d = (dst)s;
   cout << d.x;
}
```

What statement concerning the security of the program is true:

A. Bad casting gives access to private data
B. Object oriented programming is always secure
C. Casting from srs to dst is not permitted
D. Operator overloading is always secure
E. There is no operator << overloading, but it is called

Question 10 of 20
Reducing the risk of an attack on a program "managed code" is done by:

A. Insertion of additional validations
B. Use of exceptions mechanism
C. The use of privileged accounts
D. Running under a virtual machine or using code obfuscation
E. Impersonation

Question 11 of 20

```
#include <stdio.h>
void main() {
   char *pt = "01234567890";
   puts(pt + 2);
   getchar();
}
```

Displays:

A. 1234567892
B. 01234567892
C. 23456789
D. 234567890
E. 123456789

Question 12 of 20

```
namespace RaceConditions_Thread
{
        class ProcessData
        {
                int result=0;
                public void Process()
                {
                        Thread worker1=new Thread(Work1),
                                worker2=new Thread(Work2);
                                worker1.Start();
                                worker2.Start();
                                Console.Writeline(result);
                }
                void Work1(){result=1;}
                void Work2(){result=2;}

                public static void Main(string[] args){
                        ProcessData p=new ProcessData();
                        p.Process();Console.Read();
                }
        }
}
```

Displays:

A. 0, initialization value for result
B. 1, first thread assigned the result
C. 2, last thread assigned the result
D. 0, 1, 2 depending of running context (threads scheduling)

E. A residual value

Question 13 of 20

```
#include <stdio.h>
#include <string.h>
void main() {
        int count;
        printf("1234567890123456%n7890", &count);
        printf("\n %d", count);
        getchar();
}
printf("\n %d", count);
```

displays:

A. 20
B. The address of count variable
C. 16
D. 1
E. Other value

Question 14 of 20

```
#include <stdio.h>
void main() {
        char buf[] = "%p %p %p %p %p %p %p %p";
        printf(buf);
        getchar();
}
```

The sequence above:

A. Is wrong, because the printf call doesn't get the formatting information
B. Is wrong because the printf call doesn't get the variables for displaying
C. Displays variables or addresses located on the stack
D. Displays the variables or addresses from heap
E. Displays the variables or addresses from registries

Question 15 of 20

The technique for hiding information in a program which allow identify the way that an unauthorized user has come into the program possession, is called:

A. RSA Encryption
B. Assembly Linking
C. Watermarking
D. Library Bypass Obfuscation
E. Code managing

Question 16 of 20

What displays the program below, guided you by the comment in the source text

```
#include <stdio.h>
void function(int a, int b, int c) {
        char buffer1[5] = "ABCD";
```

```
        char buffer2[10] = "123456789";
        char *ret;
        ret = buffer1 + 16;
        //ret points on adr where the return address is kept
        (*ret) += 8;
}
int main() {
        int x;
        x = 0;
        function(1, 2, 3);
        x = 1;
        printf("%d\n", x);
        getchar();
        return 0;
}
```

A. 1
B. 16
C. 0
D. 8
E. 4

Question 17 of 20
Which of the following options IS NOT a remedy to SQL Injection vulnerability?

A. Writing between single quotation marks the variable part, given by the user
B. Using of stored procedures
C. Avoiding the query of remote servers
D. Using the parameters for transmission the values to server
E. Validation of the values entered by user

Question 18 of 20
Stack inspection is a technique by which:

A. It is prevented the call of a privileged function by a non privileged function
B. It is eliminated the possibility that a non privileged function to get special access rights calling a privileged function
C. Caller function is protected from the vulnerabilities of unsafe code from the called function
D. Variables or addresses on the stack are displayed
E. The data on the stack are coded for not being consulted by a malicious code

Question 19 of 20
What is the name of the arbitrary code call made by a hacker through exploiting the context of security from an assembly level?

A. Direct Manipulation
B. Virtual Attack
C. Confused Deputies
D. Code Injection
E. Spoofing

Question 20 of 20
Which of the following statements are not related to ‘strong type’ concept?

A. Clear delimitation of operations that are allowed on a type of data
B. Defining all accepted conversions for a data type
C. The knowledge of types since compilation stage
D. Initializing all variables before use
E. Prohibiting any conversion from a data type to another

Question 21.

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

int main()
{
 int lg =-10;
 char buff[100]="Exam";
 bool r = lg + strlen(buff) < sizeof(buff);
 printf(r ? "Passed" : "Failed");
 getchar();

}
```

Displays:

A. Passed
B. 1;
C. False;
D. Failed
E. true.

Question 22.

Obfuscation technique is applied for:
1. source code; 2. source code brought into intermediate language; 3.executable code.

A.1+2
B.3
C.2
D.2+3
E.1

Question 23

```
#include <iostream>
using namespace std;

void f(int x) {
	for (int i = 0;i<3;++i)
		printf("%d", *((int*)x + i));

}

void main() {
	int a = 1, b, c;
	a = (b = 2, c = 3, a<0) ? (b -= 1, b - 1) : (c -= 2, c + 2);
	cout << a << " " << b << " " << c;

	int vect[] = { 1,2,3 };
	int i = int(vect);f(i);
	getchar();

}
```

A.123123

B.332211

C.321123

D.321321

E.123321

Question 11 of 20

```
namespace RaceCondition
{
class Threads
{
int result = 0;
AutoResetEvent event1 = new AutoResetEvent(false);
AutoResetEvent event2 = new AutoResetEvent(false);
AutoResetEvent event3 = new AutoResetEvent(false);

public void Process()
{
Thread worker1 = new Thread(Work1);
Thread worker2 = new Thread(Work2);
Thread worker3 = new Thread(Work3);

WaitHandle[] waitHandles = new WaitHandle[]
{ event2, event3 };

worker1.Start(); worker2.Start(); worker3.Start();

WaitHandle.WaitAny(new WaitHandle[] { event1 });
Console.WriteLine(result);
}

void Work1()
{
WaitHandle.WaitAll(new WaitHandle[] { event2, event3 });
result = 1; event1.Set();
}

void Work2() { result = 2; event2.Set(); }
void Work3() { result = 3; event3.Set(); }

static void Main(string[] args)
{
Threads proces = new Threads();
proces.Process(); Console.Read();
}
}

}
```

displays:

- A. 0, initialization value for result
- B. 2 or 3 depending of running context(threads scheduling)
- C. 1, because first thread is the last one setting result
- D. a residual value
- E. 2, last thread assigned result

Reset Selection

Question 14 of 20

```
string frazaSQL = "select * from client where nume = " + tbNume.Text
string frazaSQL = "select * from client where nume = '" + tbNume.Text + "'";
```

Which of the two Select sentences above, you appreciate as more secure ?

- A. the first, because it not allows you to add new conditions;
- B. both are equally safe;
- C. the first, because it is easier (uncomplicated);
- D. the second, because it not allows you to add new conditions;
- E. the second because it makes validations

Reset Selection

Question 17 of 20

```
#include "stdafx.h"
#include <stdio.h>

void main()
{
int *v, b=1, c=2, n=60; v = new int[n];
for (int i = 0; i < n-2; i+=3)
v[i] = i, v[i+1] = i+1, v[i+2] = i+2;

n%3 ? (n%3==1 ? v[n-1] = n-1 : (v[n-2] = n-2), (v[n-1] = n-1) ):n;
printf("\n v[%d] = %d", n-1, v[n-1] ); getchar();
}
```

- A. is a double loop unrolling working only for n = 3k;
- B. is a double loop unrolling working only for n = 3k+1;
- C. is a double loop unrolling working for any integer n;
- D. attempts to optimize a program's execution, but doesn't obfuscate;
- E. contains sintax errors

Reset Selection

Question 17

```
#include "stdafx.h"
#include <stdio.h>

void main()
{
int *v, b=1, c=2, n=60; v= new int[n];
for (int i=0; i<n-2;i+=3)
v[i]=i, v[i+1]=i+1, v[i+2]=i+2;
n%3 ?(n%3==1 ?v[n-1] = n-1 : (v[n-2] = n-2), (v[n-1] = n-1)):n;
printf("\n v[%d]= %d", n-1, v[n-1]); getchar();
```

A. is a double loop unrolling working only for n=3k;
B. is a double loop unrolling working only for n=3k+1;
C.is a double loop unrolling working for any integer n;
D.attempts to optimize a program's execution, but doesn't Obfuscate;
E.contains sintax errors

https://www.qfeast.com/scored/quiz/UsnWyH/SourceCodeSecurity
SCS-quiz