Computer Systems and Programming

November 4th 2019

Docente: Giorgio Richelli

Aswer to questions marking the aswer that you consider as correct.
Beware: a wrong aswer will lower your score.
Time to complete the form: **60 minutes**.

Gruppi: 2/7,1/4,1/3,1/4,1/4,1/3,1/4,2/8

1. The following code fragment

```
#include <stdio.h>
float v = 1.0;
int main()
{
    int v = v;
    printf("%d\n", v);
    return 0;
}
```

(A) Prints **1** $[-WRONG]$
(B) Outputs a random value $[+CORRECT]$
(C) Generates a compiler error $[-WRONG]$
(D) Generates a compiler warning $[-WRONG]$

2. The following code fragment

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

(A) Prints **10** $[-WRONG]$
(B) Generates a compiler warning $[-WRONG]$
(C) Generates a compiler error $[+CORRECT]$
(D) Outputs a random value $[-WRONG]$

3. The following declarations

```
(1) int var_9 = 1;
(2) int 9_var = 2;
(3) int _9 = 3;
(4) int aa = 3;
}
```

(A) Are all valid [$-WRONG$]
(B) Only (4) is valid [$-WRONG$]
(C) Only (1), (3) and (4) are valid [$+CORRECT$]
(D) Only (1) and (4) are valid [$-WRONG$]

4. The snippet

```
#include "stdio.h"
int *Ptr1;
int main()
{
 int *Ptr2 = NULL;
 if(Ptr1 == Ptr2) printf("Equal");
 else printf("Not Equal");
 return 0;
}
```

(A) Prints **Equal** [$+CORRECT$]
(B) Prints **Not Equal** [$-WRONG$]
(C) It might print both [$-WRONG$]
(D) Causes a compiler error [$-WRONG$]

5. The snippet

```
#include <stdio.h>
int main()
{
int i;
  for ( i=0; i<5; i++ ) {
     int i = 10;
     printf ( "%d ", i );
     i++;
  }
  return 0;
}
```

(A) Prints **10 10 10 10 10** [$+CORRECT$]
(B) Generates a compiler error [$-WRONG$]
(C) Prints **0 1 2 3 4** [$-WRONG$]
(D) Prints **10 11 12 13 14** [$-WRONG$]

6. The snippet

```
#include <stdio.h>
int main( )
{
int i;
  for ( i=0; i<5; i++ ) {
     static int i = 10;
     printf ( "%d ", i );
     i++;
  }
  return 0;
}
```

(A) Generates a compiler error $[-WRONG]$
(B) Prints **0 1 2 3 4** $[-WRONG]$
(C) Prints **10 10 10 10 10** $[-WRONG]$
(D) Prints **10 11 12 13 14** $[+CORRECT]$

7. The snippet

```
#include <stdio.h>
int tmp=20;
void func()
{
    static int tmp=10; printf("%d ",tmp);
}
int main()
{
    printf("%d ",tmp); func(); printf("%d ",tmp);
}
```

(A) Prints **10 10 10** $[-WRONG]$
(B) Prints **20 10 10** $[-WRONG]$
(C) Prints **20 10 20** $[+CORRECT]$
(D) Prints **20 20 20** $[-WRONG]$

8. The following code fragment

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

(A) Assigns to **i** the value **1** $[-WRONG]$
(B) Assegns to **i** the value of **3** $[-WRONG]$
(C) Does not compile $[+CORRECT]$

(D) Assigns to **i** a random value $[-WRONG]$

9. The following code fragment

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

(A) Prints the value **1** $[-WRONG]$
(B) Prints the value **3** $[+CORRECT]$
(C) Does not compile $[-WRONG]$
(D) Prints a random value $[-WRONG]$

10. The following code fragment

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

(A) Prints **1** $[+CORRECT]$
(B) Prints **3** $[-WRONG]$
(C) Does not compile $[-WRONG]$
(D) Prints a random value $[-WRONG]$

11. The following code fragment

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

(A) Prints **1** $[-WRONG]$
(B) Prints **3** $[+CORRECT]$
(C) Does not compile $[-WRONG]$
(D) Prints a random value $[-WRONG]$

12. The following code fragment

```
#include <stdio.h>
int foo(int* a, int* b)
{
    int c = *a + *b;
    *b = *a;
    return *a = c - *b;
}
int main()
{
    int i = 0, j = 1, k = 2, l;
    l = i++ || foo(&j, &k);
    printf("%d %d %d %d", i, j, k, l);
    return 0;
}
```

(A) Prints **1 2 1 1** $[+CORRECT]$
(B) Prints **1 1 2 1** $[-WRONG]$
(C) Prints **1 2 2 1** $[-WRONG]$
(D) Prints **1 2 2 2** $[-WRONG]$

13. The output from the following code fragment

```
#include <stdio.h>
int f1() { printf ("AA"); return 1;}
int f2() { printf ("BB"); return 1;}

int main()
{
  int p = f1() + f2();
  return 0;
}
```

is:
(A) Compiler dependent $[+CORRECT]$
(B) **AABB** $[-WRONG]$
(C) **BBAA** $[-WRONG]$
(D) A compiler error $[-WRONG]$

14. The output from the following code fragment

```
#include <stdio.h>

int main()
{
   int a = 1, b = 1;
   int c = a || --b;
   int d = a-- && --b;
   printf("%d, %d, %d, %d", a, b, c, d);
   return 0;
}
```

is:
(A) 0, 0, 1, 0 $[+CORRECT]$
(B) 0, 1, 1, 0 $[-WRONG]$
(C) 1, 0, 1, 0 $[-WRONG]$
(D) 0, 0, 1, 1 $[-WRONG]$

15. The following code outputs

```
# include <stdio.h>
void print(int arr[])
{
   int n = sizeof(arr)/sizeof(arr[0]);
   for (int i = 0; i < n; i++)
      printf("%d ", arr[i]);
}
int main()
{
   int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
   print(arr);
   return 0;
}
```

(A) 1 2 $[-WRONG]$
(B) 1 2 3 4 5 6 7 8 $[-WRONG]$
(C) It is architecture dependent $[+CORRECT]$
(D) Generates a compiler error $[-WRONG]$

16. The following code outputs

```
#include<stdio.h>

int main()
{
  int a[] = {1, 2, 3, 4, 5, 6};
  int *ptr = (int*)(&a+1);
  printf("%d ", *(ptr-1) );
  return 0;
}
```

(A) **1** $[-WRONG]$
(B) **6** $[+CORRECT]$
(C) A compiler error $[-WRONG]$
(D) A runtime error $[-WRONG]$

17. The following code fragment

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

int main()
{
  char *a = "123456";
  char *ptr = a;
  *ptr = a[strlen(a)-1];
  printf("%s\n",a);
  return 0;
}
```

(A) Prints **523456** $[-WRONG]$
(B) Prints **623456** $[-WRONG]$
(C) Generates a compiler error $[-WRONG]$
(D) Generates a runtime error $[+CORRECT]$

18. The following code fragment

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

int main()
{
  char a[] = "123456";
  char *ptr = a;
  *ptr = a[strlen(a)-1];
  printf("%s\n",a);
  return 0;
}
```

(A) Prints **523456** $[-WRONG]$
(B) Prints **623456** $[+CORRECT]$
(C) Generates a compiler error $[-WRONG]$
(D) Generates a runtime error $[-WRONG]$

19. The function defined in the following snippet

```
int f(const int i) {
  return (i+1);
}
```

(A) Returns the value of the variable `i` [$-WRONG$]
(B) Returns the value of the variable `i` incremented by one [$+CORRECT$]
(C) Doesn't compile [$-WRONG$]
(D) Generate a warning from compiler [$-WRONG$]

20. The function defined in the following snippet

```
int f(int i) {
  if (i) return (i+1);
}
```

(A) Returns the value of the variable `i` [$-WRONG$]
(B) Returns the value of the variable `i+1` [$-WRONG$]
(C) Might return a random value [$+CORRECT$]
(D) Generates a compiler error [$-WRONG$]

21. The function defined in the following snippet, when called with a value of i=2

```
int f(const int i) {
  if (i==0) return 1;
  return (i*f(i-1));
}
```

(A) Returns a random value [$-WRONG$]
(B) Returns **2** [$+CORRECT$]
(C) Generates a runtime error [$-WRONG$]
(D) Generates a compiler error [$-WRONG$]

22. The function defined in the following snippet of code, when called with i=1

```
int f(int i) {
    return(i ? i/(i-1): -1);
}
```

(A) Returns the value **-1** [$-WRONG$]
(B) Returns the value **0** [$-WRONG$]
(C) Does not compile [$-WRONG$]
(D) Generates a runtime error [$+CORRECT$]

23. The following code fragment

```
  #include<stdio.h>
  int main()
  {
    int i=2;
    if (i!=0) {
      int i=1;
```

```
      printf("%d\n",i);
    }
  else {
    printf("%d\n",i);
  }
  return 0;
}
```

(A) Prints the value: **1** $[+CORRECT]$
(B) Prints the value: **2** $[-WRONG]$
(C) Prints a random value $[-WRONG]$
(D) Does not compile $[-WRONG]$

24. The following code fragment

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

(A) Prints the value: **3** $[-WRONG]$
(B) Prints the value: **2** $[-WRONG]$
(C) Prints a random value $[-WRONG]$
(D) Does not compile $[+CORRECT]$

25. The following fragment of code

```
#include<stdio.h>
int main()
{
  int i=2;
  if (i!=0) {
   int i;
   printf("%d\n",i);
  }
  else {
   printf("i=0\n");
  }
}
```

(A) Prints the string **i=0** $[-WRONG]$
(B) Prints the value: **2** $[-WRONG]$

(C) Prints a random value [$+CORRECT$]
(D) Does not compile [$-WRONG$]

26. The following snippet of code

```
#include<stdio.h>
int main()
{
  int i=1;
  switch(i) {
    case 1:  i= i+1;
    case 2:  i= i+1;
    default: printf("%d\n",i);
  }
}
```

(A) Prints the value: **2** [$-WRONG$]
(B) Prints the value: **3** [$+CORRECT$]
(C) Prints a random value [$-WRONG$]
(D) Does not print anything [$-WRONG$]

27. The following fragment of code

```
#include<stdio.h>
int main()
{
  int i=1;
  switch(i) {
    case 1: i= i+1; break;
    case 2: i= i+1; break;
    default: i=i+1;
  }
  printf("%d\n",i);
}
```

(A) Prints the value: **2** [$+CORRECT$]
(B) Prints the value: **3** [$-WRONG$]
(C) Prints a random value [$-WRONG$]
(D) Does not compile [$-WRONG$]

28. The following fragment of code

```
#include<stdio.h>
int main()
{
  int i=1;
  switch(i) {
```

```
    case 2:  i= i+1;
    default: printf("%d\n",i);
    case 1:  i= i+1;
  }
}
```

(A) Prints the value: **2** $[-WRONG]$
(B) Prints the value: **3** $[-WRONG]$
(C) Does not print anything $[+CORRECT]$
(D) Does not compile $[-WRONG]$

29. The following code snippet

```
#include<stdio.h>
int main()
{
  int i=3;
  switch(i) {
    case 2:  i= i+1;
    default: i= i+1;
    case 1:  i= i+1;
  }
  printf("%d\n",i);
}
```

(A) Prints the value: **3** $[-WRONG]$
(B) Prints the value: **4** $[-WRONG]$
(C) Prints the value: **5** $[+CORRECT]$
(D) Does not compile $[-WRONG]$

30. The amount of memory required to store a pointer
(A) Depends on the type of the referenced variable $[-WRONG]$
(B) It is determined only by CPU $[-WRONG]$
(C) Doesn't depend on anything (it is a constant) $[-WRONG]$
(D) It could depend by both OS and CPU $[+CORRECT]$

31. The following snippet of code

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

(A) Prints the value:**1** $[-WRONG]$
(B) Causes a runtime error $[+CORRECT]$
(C) Prints a random value $[-WRONG]$
(D) Prints the address of variable **i** $[-WRONG]$

32. The following code snippet

```
#include<stdio.h>
int main()
{
  int i=1;
  int *p=&i;
  printf("%d\n",*p);
}
```

(A) Prints the value: **1** $[+CORRECT]$
(B) Throws a runtime error $[-WRONG]$
(C) Prints a random value $[-WRONG]$
(D) Prints the address of variable **i** $[-WRONG]$

33. The following code snippet

```
#include<stdio.h>
int main()
{
  int i=1;
  int *p=&i;
  printf("%p\n",p);
}
```

(A) Prints the value: **1** $[-WRONG]$
(B) Throws a runtime error $[-WRONG]$
(C) Prints a random value $[-WRONG]$
(D) Prints the address of variable **i** $[+CORRECT]$

34. The following code snippet

```
#include <stdio.h>
int main()
{
    int x = 5;
    int const * ptr = &x;
    ++(*ptr);
    printf("%d", x);

    return 0;
}
```

(A) Prints the value: **6** $[-WRONG]$
(B) Prints the value: **5** $[-WRONG]$
(C) Causes a runtime error $[-WRONG]$
(D) Causes a compiler error $[+CORRECT]$

35. The following code snippet

```
#include <stdio.h>
int main()
{
    int x = 5;
    int * const ptr = &x;
    ++(*ptr);
    printf("%d", x);

    return 0;
}
```

(A) Prints the value: **6** $[+CORRECT]$
(B) Prints the value: **5** $[-WRONG]$
(C) Throws a runtime error $[-WRONG]$
(D) Throws a compiler error $[-WRONG]$

36. The following code snippet

```
int i=5;
#include <stdio.h>
int main()
{
  if(--i){
    main();
    printf("%d ",i);
  }
}
```

(A) Prints: **0 0 0 0** $[+CORRECT]$
(B) Prints: **4 3 2 1** $[-WRONG]$
(C) Prints: **1 2 3 4** $[-WRONG]$
(D) Throws a compiler error $[-WRONG]$

37. The following code snippet

```
#include <stdio.h>
int i=4;
int f() {
  return i--;
}
```

```
int main() {
  for(f(); f(); f()) printf("%d ", f());
  return 0;
}
```

(A) Prints: **2** [$+CORRECT$]
(B) Prints: **3** [$-WRONG$]
(C) Throws a compiler error [$-WRONG$]
(D) Never terminates (infinite loop) [$-WRONG$]