C PROGRAMMING EXAM 3

1)Write a program to ask user to enter integer and check if the number is greater than 5 print on the screen **the number is greater than or equal 5**. Do not use " if " or " switch " ;) think how to execute it.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int x=5;

    for(;x<=5;x++)
    {
        printf("the number is greater than or equal 5");
    }
    return 0;
}
```

2) How many times "my name is" will be printed.

```c
#include<stdio.h>
int main()
{
    int x;
    for(x=-1; x<=10; x++)
    {
        if(x < 5)
            continue;
        else
            break;
        printf("my name is ");
    }
    return 0;
}
```


ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

A. Infinite times

B. 11 times

C. 0 times

D. 10 times

Answer: Option **C**

3) How many times the `while` loop will get executed if a `short int` is 2 byte wide?

```
#include<stdio.h>
int main()
{
    int j=1;
    while(j <= 255)
    {
        printf("%c %d\n", j, j);
        j++;
    }
    return 0;
}
```

A. Infinite times

B. 255 times

C. 256 times

D. 254 time

Answer: Option **B**
Explanation:

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

The `while(j <= 255)` loop will get executed 255 times. The size short int(2 byte wide) does not affect the `while()` loop.

4) Which of the following cannot be checked in a `switch-case` statement?

A. Character

B. Integer

C. Float

D. enum

Answer: Option **C**
Explanation:
The `switch/case` statement in the c language is defined by the language specification to use an `int` value, so you can not use a `float` value.

```
switch( expression )
{
    case constant-expression1:    statements 1;
    case constant-expression2:    statements 2;
    case constant-expression3:    statements3 ;
    ...
    ...
    default : statements 4;
}
```

The value of the '`expression`' in a switch-case statement must be an integer, char, short, long. Float and double are not allowed.

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

1. What will be the output of the program?

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

A. 0, 1, 2, 3, 4, 5

B. 5

C. 1, 2, 3, 4

D. 6

Answer: Option **D**
Explanation:
Step 1: `int i = 0;` here variable `i` is an integer type and initialized to '0'.
Step 2: `for(; i<=5; i++);` variable `i=0` is already assigned in previous step. The semi-colon at the end of this `for` loop tells, "there is no more statement is inside the loop".

Loop 1: here `i=0`, the condition in for(; 0<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 2: here `i=1`, the condition in for(; 1<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 3: here `i=2`, the condition in for(; 2<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 4: here `i=3`, the condition in for(; 3<=5; i++) loop satisfies and then `i` is increemented by '1'(one)
Loop 5: here `i=4`, the condition in for(; 4<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 6: here `i=5`, the condition in for(; 5<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 7: here `i=6`, the condition in for(; 6<=5; i++) loop fails and then `i` is not incremented.
Step 3: `printf("%d", i);` here the value of `i` is 6. Hence the output is '6'.

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

2. What will be the output of the program?

```c
#include<stdio.h>
int main()
{
    int a = 500, b = 100, c;
    if(!a >= 400)
        b = 300;
    c = 200;
    printf("b = %d c = %d\n", b, c);
    return 0;
}
```

A. b = 300 c = 200

B. b = 100 c = garbage

C. b = 300 c = garbage

D. b = 100 c = 200

Answer: Option **D**
Explanation:
Initially variables `a = 500`, `b = 100` and `c` is not assigned.
Step 1: `if(!a >= 400)`
Step 2: `if(!500 >= 400)`
Step 3: `if(0 >= 400)`
Step 4: `if(FALSE)` Hence the `if` condition is failed.
Step 5: So, variable `c` is assigned to a value '200'.
Step 6: `printf("b = %d c = %d\n", b, c);` It prints value of `b` and `c`.
Hence the output is "b = 100 c = 200"

3. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    unsigned int i = 65535; /* Assume 2 byte integer*/
    while(i++ != 0)
        printf("%d",++i);
    printf("\n");
    return 0;
}
```

A. Infinite loop

B. 0 1 2 ... 65535

C. 0 1 2 ... 32767 - 32766 -32765 -1 0

D. No output

Answer: Option **A**
Explanation:
Here `unsigned int` size is 2 bytes. It varies from 0,1,2,3, ... to 65535.
Step 1:`unsigned int i = 65535;`
Step 2:
Loop 1: `while(i++ != 0)` this statement becomes `while(65535 != 0)`. Hence the `while(TRUE)` condition is satisfied. Then the `printf("%d", ++i);` prints '1'(variable `'i'` is already incremented by '1' in while statement and now incremented by '1' in printf statement) **Loop 2**: `while(i++ != 0)` this statement becomes `while(1 != 0)`. Hence the `while(TRUE)` condition is satisfied. Then the `printf("%d", ++i);` prints '3'(variable `'i'` is already incremented by '1' in while statement and now incremented by '1' in printf statement)
....
....
The while loop will never stops executing, because variable `i` will never become '0'(zero). Hence it is an 'Infinite loop'.

ENG. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

5. What will be the output of the program?

```c
#include<stdio.h>
int main()
{
    int x = 3;
    float y = 3.0;
    if(x == y)
        printf("x and y are equal");
    else
        printf("x and y are not equal");
    return 0;
}
```

A. x and y are equal

B. x and y are not equal

C. Unpredictable

D. No output

Answer: Option **A**
Explanation:
Step 1: `int x = 3;` here variable `x` is an integer type and initialized to '3'.
Step 2: `float y = 3.0;` here variable `y` is an float type and initialized to '3.0'
Step 3: `if(x == y)` here we are comparing `if(3 == 3.0)` hence this condition is satisfied. Hence it prints "x and y are equal".