Assignment 1

J. Prabhath - EE18BTECH11021

Download all Codes from

```
https://github.com/jpln135/EE4013/tree/master/
    Assignment1/codes
```

Download all latex-tikz codes from

```
https://github.com/jpln135/EE4013/tree/master/
    Assignment1/Assignment1.tex
```

1 Problem

Consider the following C code

```
#include <stdio.h>
int * assignval (int *x, int val)
{
    *x = val;
    return x;
}
int main()
{
    int *x = malloc(sizeof(int));
    if (NULL == x) return;
    x = assignval(x, 0);
    if(x)
    {
        x = (int*) malloc(sizeof (int));
        if (NULL == x) return;
        x = assignval (x, 10);
    }
    printf("%d\n", *x);
    free(x);
}
```

The code suffers from which of the following problems:
(A) compiler error as the return of malloc is not typecast appropriately.
(B) compiler error because the comparison should be made as x==NULL and not as shown.
(C) compiles successfully but execution may result in dangling pointer.
(D) compiles successfully but execution may result in memory leak.

2 Solution

Answer: (D) compiles successfully but execution may result in memory leak.

Output:
10

Memory leak occurs when memory is created in the heap but not deleted.

Consider the line of code,

```
int *x = malloc(sizeof(int));
```

In the above line of code, 4-bytes of memory (Size of integer) is created and the address is assigned to x;

Consider the line of code (in if-block),

```
x = (int*) malloc(sizeof (int));
```

In the above line of code, pointer variable is reassigned to a new address loacation.

From the above 2 lines we can conclude that, the initial address to which the pointer variable x was pointing to is lost and hence a memory leak is present.

We can correct this by freeing the memory space before allocating new memory.

```
#include <stdio.h>
int * assignval (int *x, int val)
{
    *x = val;
    return x;
}
int main()
{
    int *x = malloc(sizeof(int));
    if (NULL == x) return;
    x = assignval(x, 0);
    if(x)
    {
        free(x); // Free memory
        x = (int*) malloc(sizeof (int));
        if (NULL == x) return;
        x = assignval (x, 10);
    }
    printf("%d\n", *x);
    free(x);
}
```

Output:

```
10
```