

# For Next Day 2

## 65 Printed as specified

### Code

```
//This program prints out the integer value 65 as %d, %4d, %x, %o, and %c|
//@author Mahmoud Moustafa; ID:3648276
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char * * argv)
{
    printf("65 printed with %d is %d\n", 65); //65 is printed as a decimal integer
    printf("65 printed with %4d is %4d\n", 65); //65 is printed as a decimal integer with a width of 4
    printf("65 printed with %x is %x\n", 65); //65 is printed as a hexadecimal
    printf("65 printed with %o is %o\n", 65); // 65 is printed as an octal
    printf("65 printed with %c is %c\n", (char)65); //65 is printed as a char

    return EXIT_SUCCESS;
}
```

### Output

```
[mmoustaf@gc112m38 ~]$ cd cs2263/
[mmoustaf@gc112m38 ~/cs2263]$ cd forNextDay/
[mmoustaf@gc112m38 forNextDay]$ cd F
FND1/ FND2/
[mmoustaf@gc112m38 forNextDay]$ cd FND2
[mmoustaf@gc112m38 FND2]$ gcc -o integer
declare.c  integer.c  playStack.c
[mmoustaf@gc112m38 FND2]$ gcc -o integer integer.c
[mmoustaf@gc112m38 FND2]$ ./integer
65 printed with %d is 65
65 printed with %4d is   65
65 printed with %x is 41
65 printed with %o is 101
65 printed with %c is A
[mmoustaf@gc112m38 FND2]$
```

## Sizeof

### Code

```
//This program prints the size of different data types
//@author Mahmoud Moustafa; ID:3648276
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char * * argv)
{
    char c;
    int i;
    float f;
    double d;
    printf("The size of char is %d byte\n", sizeof(c));
    printf("The size of int is %d bytes\n", sizeof(i));
    printf("The size of float is %d bytes\n", sizeof(f));
    printf("The size of double is %d bytes\n", sizeof(d));

    return EXIT_SUCCESS;
}
```

### Output

```
[mmoustaf@gc112m38 FND2]$ gcc -o declare declare.c
[mmoustaf@gc112m38 FND2]$ ./declare
The size of char is 1 byte
The size of int is 4 bytes
The size of float is 4 bytes
The size of double is 8 bytes
[mmoustaf@gc112m38 FND2]$
```

### What these values mean

These values are the storage/memory space that different datatypes occupy.

The minimum and maximum values for a signed int.

Terminal

```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\momou> ssh mmoustaf@gc112m38.cs.unb.ca
mmoustaf@gc112m38.cs.unb.ca's password:
Last login: Sat Sep 11 16:02:59 2021 from 10.6.104.12
[mmoustaf@gc112m38 ~]$ cd ..
[mmoustaf@gc112m38 ugrads]$ cd ..
[mmoustaf@gc112m38 /home1]$ cd ..
[mmoustaf@gc112m38 /]$ cd ..
[mmoustaf@gc112m38 /]$ cd usr
[mmoustaf@gc112m38 /usr]$ cd include/
[mmoustaf@gc112m38 include]$ vi li
libdb/      libgen.h  libio.h    libpng15/  libyami/   link.h
libdrm/     libintl.h  libkms/    libsync.h  limits.h   linux/
[mmoustaf@gc112m38 include]$ vi limits.h
```

```
/* Minimum and maximum values a `signed int' can hold. */
# define INT_MIN      (-INT_MAX - 1)
# define INT_MAX      2147483647
```

Minimum value for a signed int: -2147483648

Maximum value for a signed int: 2147483647

# playStack

## Code

```
1 // first.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define MAX 256
5 #define PUSH 1
6 #define POP 0
7 #define LIST 2
8 int main(int argc, char* argv[])
9 {
10     int stack[MAX];
11     int size = 0;
12     int val;
13     int ichoice;
14     int iNRead;
15     int counter;
16
17     /* Processing loop */
18     printf("Choice (1=add, 0=remove, 2=list): ");
19     iNRead = scanf("%d", &ichoice);
20     while(iNRead == 1)
21     {
22         switch(ichoice)
23         {
24             case PUSH:
25                 printf("Value to add: ");
26                 // Read the element, add it to the stack
27                 scanf("%d", &val);
28                 if (size < MAX) {
29                     stack[size] = val;
30                     size++;
31                 }
32                 break;
33             case POP:
34                 // Print out the last element and remove it.
35                 if (size == 0)
36                 {
37                     printf("There is nothing in the stack");
38                 }
39             }
40     }
41 }
```

```
33     case POP:
34         // Print out the last element and remove it.
35         if (size == 0)
36         {
37             printf("There is nothing in the stack");
38         }
39         printf("%d\n", stack[size-1]);
40         size--;
41         break;
42     case LIST:
43         // Print out the stack elements
44         counter = size;
45         if (size == 0)
46         {
47             printf("The stack is empty");
48         }
49         else
50         {
51             while (counter > 0)
52             {
53                 printf("%d ", stack[counter-1]);
54                 counter--;
55             }
56             break;
57         }
58     }
59     printf("Choice (1=add, 0=remove, 2=list): ");
60     iNRead = scanf("%d", &ichoice);
61 }
62 return EXIT_SUCCESS;
63 }
64 }
```

## Terminal Output

```
mmoustaf@gc112m38:FND2
[mmoustaf@gc112m38 FND2]$ gcc -o playStack playStack.c
[mmoustaf@gc112m38 FND2]$ ./playStack
Choice (1=add, 0=remove, 2=list): 1
Value to add: 1
Choice (1=add, 0=remove, 2=list): 1
Value to add: 2
Choice (1=add, 0=remove, 2=list): 1
Value to add: 3
Choice (1=add, 0=remove, 2=list): 1
Value to add: 4
Choice (1=add, 0=remove, 2=list): 1
Value to add: 5
Choice (1=add, 0=remove, 2=list): 2
5 4 3 2 1
Choice (1=add, 0=remove, 2=list): 1
Value to add: 6
Choice (1=add, 0=remove, 2=list): 2
6 5 4 3 2 1
Choice (1=add, 0=remove, 2=list): 0
6
Choice (1=add, 0=remove, 2=list): 2
5 4 3 2 1
Choice (1=add, 0=remove, 2=list): 0
5
Choice (1=add, 0=remove, 2=list): 0
4
Choice (1=add, 0=remove, 2=list): 0
3
Choice (1=add, 0=remove, 2=list): 2
2 1
Choice (1=add, 0=remove, 2=list):
```