भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी

Indian Institute of Information Technology Guwahati

# COMPUTER PROGRAMMING LAB (CS110)
# ASSIGNMENTS AND SOLUTIONS–01

1. Write a program to print the message `"Hello, world!"`, which needs to be followed by a newline character.

```c
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

2. Try to declare variables of type `int` with the following variable names:

   i. `my_first_variable`
   ii. `mySecondVariable`
   iii. `MyThirdVariable`
   iv. `char`
   v. `n`
   vi. `number`
   vii. `_`
   viii. `_number_`
   ix. `2months`
   x. `months2`
   xi. `months_2`
   xii. `months 2`
   xiii. `months two`
   xiv. `months?`

   *Hint:* Cases iv, ix, xii, xiii, and xiv do not work.

```c
#include <stdio.h>

int main() {
    int my_first_variable;
    int mySecondVariable;
    int MyThirdVariable;
    int char; // Error
    int n;
    int number;
    int _;
    int _number_;
    int 2months; // Error
    int months2;
    int months_2;
    int months 2; // Error
    int months two; // Error
    int months?; // Error
    return 0;
}
```

3. Decalre a variable of type `int` with the varible name `n`. Initialize `n` as 5. Print the value of `n`. Now, print the addresses of `n` (use `&`) using `%p` as the corresponding format specifier.
   *Hint:*
   ```
   int n = 5;
   printf("The value of n is %d.\n", n);
   printf("The address of n is %p.\n", &n);
   ```

```c
#include <stdio.h>

int main() {
    int n = 5;
    printf("The value of n is %d.\n", n);
    printf("The address of n is %p.\n", &n);
    return 0;
}
```

4. Print the address of the `main()` function and the `printf()` function.
   *Hint:*
   ```
   printf("The address of main() is %p.\n", main);
   printf("The address of main() is %p.\n", &main);
   printf("The address of printf() is %p.\n", printf);
   printf("The address of printf() is %p.\n", &printf);
   ```

```c
#include <stdio.h>

int main() {
    printf("The address of main() is %p.\n", main);
```

```
    printf("The address of main() is %p.\n", &main);
    printf("The address of printf() is %p.\n", printf);
    printf("The address of printf() is %p.\n", &printf);
    return 0;
}
```

5. Declare variables as follows:

```
char myChar = ‘C’;
unsigned char myUnsignedChar = ‘C’;
signed char mySignedChar = ‘C’;
int myInt = -1 * ‘C’;
unsigned int myUnsignedInt = 0x5E;
short myShort = -1 * ‘C’;
unsigned short myUnsignedShort = 010;
long myLong = -10000000;
unsigned long myUnsignedLong = 10000000000;
float myFloat = 0.325;
double myDouble = 1.5e-3;
long double myLongDouble = 3.2e30;
```

Use the `sizeof()` operator to print the size of each of the variables using the operator both on the variables and on the datatypes. Try to print each of these variables with the following format specifiers: `"%c"`, `"%d"`, `"%f"`, `"%g"`, `"%e"`, `"%lf"`, `"%o"`, `"%x"` and `"%s"`.

*Hint:*

```
int x = 2;
printf("%ld %ld %ld\n", sizeof(x), sizeof(int), sizeof(typeof(x)));
```

```
#include <stdio.h>

int main() {
    char myChar = 'C';
    unsigned char myUnsignedChar = 'C';
    signed char mySignedChar = 'C';
    int myInt = -1 * 'C';
    unsigned int myUnsignedInt = 0x5E;
    short myShort = -1 * 'C';
    unsigned short myUnsignedShort = 010;
    long myLong = -10000000;
    unsigned long myUnsignedLong = 10000000000;
    float myFloat = 0.325;
    double myDouble = 1.5e-3;
    long double myLongDouble = 3.2e30;

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
```

```
        "%%o = %o, %%x = %x\n", sizeof(myChar),
        sizeof(typeof(myChar)), myChar, myChar, myChar, myChar,
        myChar, myChar, myChar, myChar
    ); // %s may cause "Segmentation fault"

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
        "%%o = %o, %%x = %x\n", sizeof(myUnsignedChar),
        sizeof(typeof(myUnsignedChar)), myUnsignedChar,
        myUnsignedChar, myUnsignedChar, myUnsignedChar,
        myUnsignedChar, myUnsignedChar, myUnsignedChar,
        myUnsignedChar
    ); // %s may cause "Segmentation fault"

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
        "%%o = %o, %%x = %x\n", sizeof(mySignedChar),
        sizeof(typeof(mySignedChar)), mySignedChar,
        mySignedChar, mySignedChar, mySignedChar,
        mySignedChar, mySignedChar, mySignedChar,
        mySignedChar
    ); // %s may cause "Segmentation fault"

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
        "%%o = %o, %%x = %x\n", sizeof(myInt),
        sizeof(typeof(myInt)), myInt, myInt, myInt,
        myInt, myInt, myInt, myInt, myInt
    ); // %s may cause "Segmentation fault"

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
        "%%o = %o, %%x = %x\n", sizeof(myUnsignedInt),
        sizeof(typeof(myUnsignedInt)), myUnsignedInt,
        myUnsignedInt, myUnsignedInt, myUnsignedInt,
        myUnsignedInt, myUnsignedInt, myUnsignedInt,
        myUnsignedInt
    ); // %s may cause "Segmentation fault"

    printf(
        "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
        "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
        "%%o = %o, %%x = %x\n", sizeof(myShort),
        sizeof(typeof(myShort)), myShort, myShort, myShort,
        myShort, myShort, myShort, myShort, myShort
    ); // %s may cause "Segmentation fault"
```

```
printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myUnsignedShort),
    sizeof(typeof(myUnsignedShort)), myUnsignedShort,
    myUnsignedShort, myUnsignedShort, myUnsignedShort,
    myUnsignedShort, myUnsignedShort, myUnsignedShort,
    myUnsignedShort
); // %s may cause "Segmentation fault"

printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myLong),
    sizeof(typeof(myLong)), myLong, myLong, myLong,
    myLong, myLong, myLong, myLong, myLong
); // %s may cause "Segmentation fault"

printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myUnsignedLong),
    sizeof(typeof(myUnsignedLong)), myUnsignedLong,
    myUnsignedLong, myUnsignedLong, myUnsignedLong,
    myUnsignedLong, myUnsignedLong, myUnsignedLong,
    myUnsignedLong
); // %s may cause "Segmentation fault"

printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myFloat),
    sizeof(typeof(myFloat)), myFloat, myFloat, myFloat,
    myFloat, myFloat, myFloat, myFloat, myFloat
); // %s may cause "Segmentation fault"

printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myDouble),
    sizeof(typeof(myDouble)), myDouble, myDouble, myDouble,
    myDouble, myDouble, myDouble, myDouble, myDouble
); // %s may cause "Segmentation fault"

printf(
    "size = %d, size(type) = %d, %%c = %c, %%d = %d, "
    "%%f = %f, %%g = %g, %%e = %e, %%lf = %lf, "
    "%%o = %o, %%x = %x\n", sizeof(myLongDouble),
    sizeof(typeof(myLongDouble)), myLongDouble,
```

```
        myLongDouble, myLongDouble, myLongDouble,
        myLongDouble, myLongDouble, myLongDouble,
        myLongDouble
    ); // %s may cause "Segmentation fault"

    return 0;
}
```

6. Consider five variables initialized as follows:
   `int a = 7, b = 5, c = 0, d = 5, result = 0;`
   Print the value of `result` after each of the following statements:

   i. `result = a + b;`
   ii. `result = a - b;`
   iii. `result = a / b;`
   iv. `result = a * b;`
   v. `result = a % b;`
   vi. `result = a + b;`
   vii. `result++;`
   viii. `result--;`
   ix. `++result;`
   x. `--result;`
   xi. `result = (a == b);`
   xii. `result = (a != b);`
   xiii. `result = (a < b);`
   xiv. `result = (a <= b);`
   xv. `result = (d > b);`
   xvi. `result = (d >= b);`
   xvii. `result = (c && d);`
   xviii. `result = (c || d);`
   xix. `result = (a & b);`
   xx. `result = (a && b);`
   xxi. `result = (a | b);`
   xxii. `result = (a || b);`
   xxiii. `result = (a ^ b);`

xxiv. `result = (a << 2);`

xxv. `result = (a >> 2);`

xxvi. `result += a;`

xxvii. `result -= a;`

xxviii. `result *= a;`

xxix. `result /= a;`

xxx. `result >>= 2;`

xxxi. `result <<= 2;`

xxxii. `result &= 2;`

xxxiii. `result |= 2;`

xxxiv. `result = (a < b) ?  c :  d;`

xxxv. `result = (a > b) ?  c :  d;`

```c
#include <stdio.h>

int main() {
    int a = 7, b = 5, c = 0, d = 5, result = 0;
    result = a + b;
    printf("result = %d\n", result);
    result = a - b;
    printf("result = %d\n", result);
    result = a / b;
    printf("result = %d\n", result);
    result = a * b;
    printf("result = %d\n", result);
    result = a % b;
    printf("result = %d\n", result);
    result = a + b;
    printf("result = %d\n", result);
    result++;
    printf("result = %d\n", result);
    result--;
    printf("result = %d\n", result);
    ++result;
    printf("result = %d\n", result);
    --result;
    printf("result = %d\n", result);
    result = (a == b);
    printf("result = %d\n", result);
    result = (a != b);
    printf("result = %d\n", result);
    result = (a < b);
    printf("result = %d\n", result);
    result = (a <= b);
```

```
    printf("result = %d\n", result);
    result = (d > b);
    printf("result = %d\n", result);
    result = (d >= b);
    printf("result = %d\n", result);
    result = (c && d);
    printf("result = %d\n", result);
    result = (c || d);
    printf("result = %d\n", result);
    result = (a & b);
    printf("result = %d\n", result);
    result = (a && b);
    printf("result = %d\n", result);
    result = (a | b);
    printf("result = %d\n", result);
    result = (a || b);
    printf("result = %d\n", result);
    result = (a ^ b);
    printf("result = %d\n", result);
    result = (a << 2);
    printf("result = %d\n", result);
    result = (a >> 2);
    printf("result = %d\n", result);
    result += a;
    printf("result = %d\n", result);
    result -= a;
    printf("result = %d\n", result);
    result *= a;
    printf("result = %d\n", result);
    result /= a;
    printf("result = %d\n", result);
    result >>= 2;
    printf("result = %d\n", result);
    result <<= 2;
    printf("result = %d\n", result);
    result &= 2;
    printf("result = %d\n", result);
    result |= 2;
    printf("result = %d\n", result);
    result = (a < b) ? c : d;
    printf("result = %d\n", result);
    result = (a > b) ? c : d;
    printf("result = %d\n", result);
    return 0;
}
```