* **1 Mark – MCQ**

1. Which is valid C expression?  
   a) int my\_num = 100,000;  
   **b) int my\_num = 100000;**  
   c) int my num = 1000;  
   d) int $my\_num = 10000;
2. Which of the following is not a pointer declaration?  
   a) char a[10];  
   b) char a[] = {‘1’, ‘2’, ‘3’, ‘4’};  
   c) char \*str;  
   **d) char a;**
3. Which of the following data type will throw an error on modulus operation(%)?  
   a) char  
   b) short  
   c) int  
   **d) float**
4. Which of the following type-casting have chances for wrap around?  
   a) From int to float  
   **b) From int to char**c) From char to short  
   d) From char to int
5. What will be the final values of a and c in the following C statement? (Initial values: a = 2, c = 1)

c = (c) ? a = 0 : 2;

**a) a = 0, c = 0;**b) a = 2, c = 2;  
c) a = 2, c = 2;  
d) a = 1, c = 2;

1. What will be the final value of c in the following C statement? (Initial value: c = 2)

c <<=1;

a) c = 1;  
b) c = 2;  
c) c = 3;  
**d) c = 4;**

1. Which of the following fopen() statements are illegal?  
   a) fp = fopen(“abc.txt”, “r”);  
   b) fp = fopen(“/home/user1/abc.txt”, “w”);  
   c) fp = fopen(“abc”, “w”);  
   **d) none of the mentioned**
2. The toupper() function converts a\_\_\_\_\_\_to the corresponding\_\_\_\_\_\_  
   a) uppercase, lowercase  
   **b) lowercase, uppercase**  
   c) binary, decimal  
   d) decimal, binary
3. Which among the following is never possible in C when members are different in a structure and union?

//Let P be a structure

//Let P be a union

a) sizeof(P) is greater than sizeof(Q)  
b) sizeof(P) is less than sizeof(Q)  
c) sizeof(P) is equal to sizeof(Q)  
**d) none of the mentioned**

1. The number of arguments taken as input which allocating memory dynamically using malloc() is \_\_\_\_\_\_\_\_\_\_\_  
   a) 0  
   **b) 1**  
   c) 2  
   d) 3

* **2 Marks – MCQ**

1. What will be the output of the following C code?

#include <stdio.h>

int main()

{

float f1 = 0.1;

if (f1 == 0.1)

printf("equal\n");

else

printf("not equal\n");

}

1. equal  
   **b) not equal**  
   c) output depends on the compiler  
   d) none of the mentioned
2. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int a =10;

double b =5.6;

int c;

c = a + b;

printf("%d", c);

}

**a) 15**  
b) 16  
c) 15.6  
d) 10

1. What will be the output of the following C code considering the size of short int is 2, char is 1 and int is 4 bytes?

#include <stdio.h>

int main()

{

shortint i =20;

char c =97;

printf("%d, %d, %d\n",sizeof(i),sizeof(c),sizeof(c + i));

return0;

}

a) 2, 1, 2  
b) 2, 1, 1  
**c) 2, 1, 4**d) 2, 2, 8

1. Assume the following C variable declaration

int \*A [10], B[10][10]; Of the following expressions

I. A[2]  
II. A[2][3]  
III. B[1]  
IV. B[2][3]

which will not give compile-time errors if used as left hand sides of assignment statements in a C program?  
  
**a) I, II, and IV only**b) II, III, and IV only  
c) II and IV only  
d) IV only

1. What will be the output of the following C code?

#include <stdio.h>

void main()

{

int a = 5, b = -7, c = 0, d;

d = ++a && ++b || ++c;

printf("\n%d%d%d%d", a, b, c, d);

}

a) 6 -6 0 0  
b) 6 -5 0 1  
c) -6 -6 0 1  
**d) 6 -6 0 1**

1. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int x = 3; //, y = 2;

const int \*p = &x;

\*p++;

printf("%d\n", \*p);

}

a) Increment of read-only location compile error  
b) 4  
**c) Some garbage value**  
d) Undefined behaviour

1. What are the different ways to initialize an array with all elements as zero?  
   a) int array[5] = {};  
   b) int array[5] = {0};  
   c) int a = 0, b = 0, c = 0;

int array[5] = {a, b, c};

**d) All of the mentioned**

1. What will be the output of the following C code?

#include <stdio.h>

struct point

{

int x;

int y;

};

void foo(struct point\*);

int main()

{

struct point p1[] = {1, 2, 3, 4};

foo(p1);

}

void foo(struct point p[])

{

printf("%d\n", p->x);

}

**a) 1**  
b) 2  
c) 3  
d) Compile time error

1. What will be the output of the following C code?

int ch = '\t';

if(isprint(ch))

printf("ch = |%c| printable \n", ch);

else

printf("ch= |%c| not printable \n",ch);

a) ch = |\t| printable  
b) ch = |\t| not printable  
c) ch = | | printable  
**d) ch = | | not printable**

1. What is the output of the C code shown below if input given is 2?

#include<stdio.h>

enum day

{

a,b,c=5,d,e

};

main()

{

printf("Enter the value for a");

scanf("%d",a);

printf("%d",a);

}

a) 2  
b) 0  
c) 3  
**d) Error**