Module 1 Assessment

By Shailesh Babu.M

1. Write a C program to determine the given number is odd or even using Bitwise operators.

```
#include <stdio.h>
int main()
{
   int num;
   printf("Enter any number: ");
   scanf("%d", &num);
   if(num & 1)
   {
      printf("%d is odd.", num);
   }
   else
   {
      printf("%d is even.", num);
   }
   return 0;
}
```

2. Write a C program to count the number of bits set in a number.

```
#include <stdio.h>

unsigned int countSetBits(unsigned int n)

{

   unsigned int count = 0;

   while (n) {

      count += n & 1;

      n >>= 1;

   }

   return count;

}


int main()
```

```
{
    int num;
    printf("Enter any number: ");
    scanf("%d", &num);
    printf("%d", countSetBits(num));
    return 0;
}
```

3. Write a C program to swap two numbers. Use a function pointer to do this operation.

```
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main() {
    int n1, n2;
    printf("Enter two numbers: ");
    scanf("%d %d", &n1, &n2);
    void (*ptr)(int *, int *);
    ptr = swap;
    ptr(&n1, &n2);
    printf("After swapping: %d %d\n", n1, n2);
    return 0;
}
```

4. Write an equivalent pointer expression for fetching the value of array element a[i][j][k][2]

((*(*(a + i) + j) + k) + 2)

5. Write a C program to Multiply two matrix (n*n) using pointers.

```
#include <stdio.h>
void multiply(int *mat1, int *mat2, int *result, int n) {
    int i, j, k;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            *(result + i * n + j) = 0;            for (k = 0; k < n; k++) {
                *(result + i * n + j) += *(mat1 + i * n + k) * *(mat2 + k * n + j);
            }
        }
    }
}
void display(int *matrix, int n) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", *(matrix + i * n + j));
        }
        printf("\n");     }
}
int main() {
    int n;
    printf("Enter the size of the square matrices: ");
```

```
    scanf("%d", &n);
    int mat1[n][n], mat2[n][n], result[n][n];
    printf("Enter elements of matrix 1:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }
    printf("Enter elements of matrix 2:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }
    multiply((int *)mat1, (int *)mat2, (int *)result, n);
    printf("Product of the matrices:\n");
    display((int *)result, n);
    return 0;
}
```

6. Find the output of the following // Consider the compiler is 32-bit machine

The total size = 6 bytes

(4 bytes for int A + 1 byte for char B + 1 byte for char C).

The total size of the structure should be the 8 bytes.

Therefore, the output is,

Size of Structure = 8

7. Find the output of the following // Consider the compiler is 32-bit machine

The total size = 10 bytes

(1 byte for char A + 8 bytes for double B + 1 byte for char C)

The total size of the structure should be multiple of 8, which is 16 bytes.

Therefore, the output is,

Size of Structure = 16

8. Find the output of the following // Consider the compiler is 32-bit machine

(i = 0), the rightmost nibble of var (i.e., 0x8) is isolated using var >> (4 * 0), then masked with 0xF to ensure only the nibble is retained. This nibble is then appended to the left of rev using rev = (rev << 4)

(i = 1), the next nibble (0x7) is isolated and appended to the left of rev.

The process continues until all 8 nibbles are reversed.

Therefore the output is,

87654321