**SOLUTIONS MID SEMESTER 2025-2026**
**(Fundamentals of Computer and C programming)**

1a)
```
min_val = x<y ? x : y;
```
1b)
```
is_odd = x & 1;
            OR
   is_odd =  x % 2;
```

2 a) `4 3 2 1 0`

2 b) `x = 14 y=29`

2 c) `c`

2 d) `case 1 default`

3 a) Observe that `0^1^2 … ^ 255 = 0` since the bit at each position is `0` in `128` of these numbers and is `1` in the remaining `128` of them. Moreover, `252^253^254^255 = 0`, since each of these four numbers differ only in their last two bits, which are `00`, `01`, `10`, and `11` respectively. Hence,

```
1^2^3 … ^ 250 = (0^1^2 … ^ 255) ^ (251 ^ 252 ^253 ^254 ^255)
              = 251 ^ 252 ^253 ^254 ^255
              = 251 ^ 0
              = 251
```

3 b)

- Both if-else and switch statements are branching statements. If-else statements evaluate Boolean conditions sequentially and decide in which unique body the execution happens. Switch evaluates an integer condition and matches it against a list of constant integers labels. Depending on the placement of break statements within cases and the presence of default case, the body of zero, one, or more than one case-blocks may be evaluated.
- In C, a string is stored as an array of characters, which is allocated contiguous memory. To determine where the string ends within this array, C employs a convention: every string is terminated by a special

null character ('\0') with ASCII code 0. This is the signal for all C string functions to know they have reached the end of the string.

3c)

- Both break and continue are jump statements. The break statement immediately terminates the inner-most loop it is in, causing program execution to resume at the statement following that loop. In contrast, the continue statement only skips the remainder of the current iteration of the inner-most loop it is in, and program control proceeds to the next iteration of the loop, where the update condition (if applicable) and stopping condition are re-evaluated.
- In entry controlled loops (while and for loops in C), the stopping condition is evaluated before executing its body, meaning the body never runs if the condition is initially false. In exit-controlled loops (do-while loop in C) the stopping condition is evaluated after executing its body, which guarantees the loop's body will run at least once.

4 a)

```c
#include <stdio.h>
int main()
{
    long int N;
    printf("Enter a positive long int N: ");
    scanf("%ld", &N);
    if(N < 1) {
        printf("Please enter positive N. \n");
    } else if(N == 1) {
        printf("1 has no prime factors.\n");
    } else {
        printf("Prime factorization of %ld = ", N);
         //the loop counter is a candidate factor of N. It
should be long int since it may become as large as N
        for(long int f=2; f<=N; f++) {
            if(N%f == 0) {
```

```
                int k = 0; //k will store the power of f in N
                while(N%f == 0) {
                    N /= f;
                    k += 1;
                }
                printf("%ld^%d ", f, k);
            }
        }
    }
    return 0;
}
```

4 b)

```
#include <stdio.h>
int main()
{
    int N;
     //fibo should be a long int array because fibo[60] ~ pow(Phi, 60) which is larger than pow(2, 31) but smaller than pow(2, 63). Here Phi is the golden ratio ~ 1.6.
    long int fibo[60] = {0, 1};

    printf("Enter N: ");
    scanf("%d", &N);
    if(N < 1 || N > 60) {
        printf("Please enter N between 1 and 60.\n");
        return 1;
    }

    for(int i=2; i<N; i++) fibo[i] = fibo[i-1] + fibo[i-2];
    printf("%d terms of Fibonacci series are: ", N);
    for(int i=0; i<N; i++) printf("%ld ", fibo[i]);
    printf("\n");
    return 0;
}
```

4 c)

```c
#include <stdio.h>
#define MAXN 20
int main()
{
    long double arr[MAXN];
    printf("Enter %d long double values: ", MAXN);
    //%Lf is the long double type identifier
    for(int i=0; i<MAXN; i++) scanf("%Lf", &arr[i]);
    printf("The elements located at even indices in the array in
the reverse order are: ");
    //the array index starts at 0 and goes until MAXN-1
    for(int i=MAXN-1; i>=0; i--) {
        if((i%2) == 0) printf("%Lf ", arr[i]);
    }
    printf("\n");
    return 0;
}
```