

Family Name:..... Other Names:.....

Student ID:..... Signature.....

NWEN241: Mid-Term Test

31 March 2026 ** WITH SOLUTION

Instructions

- Time allowed: **120 minutes**
- There are **100** marks in total.
- Attempt **ALL** the **50** multiple choice questions. Fill in circle corresponding to your answer on the Answer Sheet.
- The Answer Sheet provides space for 60 questions. **Please fill in only items 1–50 and leave items 51–60 blank.**
- Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.
- Only paper (non-electronic) foreign to English language dictionaries are allowed.
- You **MUST** submit this test paper together with the Answer Sheet.

There are TEN sections:

A	Introduction to Systems Programming	[6 Marks]
B	C Fundamentals	[14 Marks]
C	Macros and Arrays	[10 Marks]
D	Strings	[8 Marks]
E	Structures	[8 Marks]
F	Pointers	[14 Marks]
G	Storage Classes and Process Layout	[10 Marks]
H	Dynamic Memory Management	[12 Marks]
I	User-Defined Types	[8 Marks]
J	FILE Stream I/O	[10 Marks]
	TOTAL	[100 Marks]

SECTION A Introduction to Systems Programming

1. Which of the following is **NOT** a systems program? **(2 marks)**

- (a) Android operating system
- (b) C compiler
- (c) Database server program
- (d) Spreadsheet program
- (e) Virtual machine

d

2. What is the main difference between the directives `#include <header_file>` and `#include "header_file"`? **(2 marks)**

- (a) There is no difference between the directives
- (b) `#include <header_file>` can only be used to include user-defined header files
- (c) `#include "header_file"` search the current directory first, then predefined locations
- (d) `#include <header_file>` search the current directory first, then predefined locations
- (e) None of the above

c

3. What is the correct sequence of the C compilation process? **(2 marks)**

- (a) Preprocessing, Assembly, Linking, Compilation
- (b) Preprocessing, Compilation, Assembly, Linking
- (c) Compilation, Assembly, Linking, Preprocessing
- (d) Assembly, Compilation, Preprocessing, Linking
- (e) Linking, Preprocessing, Assembly, Compilation

b

SECTION B C Fundamentals

4. Which of the following is **NOT** a valid C identifier? **(2 marks)**

- (a) `_double_`
- (b) `bool`
- (c) `Struct`
- (d) `hello123`
- (e) None of the above

b

5. What is the data type of the literal `0xabcdL`? **(2 marks)**

- (a) `int`
- (b) `unsigned long`
- (c) `long int`
- (d) `long double`
- (e) None of the above

c

6. A C program contains the following declarations: **(2 marks)**

```
int a, b;  
long c;  
short d;  
float e;  
char f;
```

What is the resulting data type of the following expression?

$$13 * f + (\text{long}) (b / d) - e * c / 2.5$$

[Hint: The C operator precedence table is on page 24.]

- (a) `short`
- (b) `int`
- (c) `long`
- (d) `float`
- (e) None of the above

e

7. Suppose a, b and c are integral type variables that have been assigned the values a = 8, b = 3 and c = 4. What value does the following expression evaluate to? (2 marks)

$a / b + 5 * (b = 5) * (a - c)$

[Hint: The C operator precedence table is on page 24.]

- (a) 2
- (b) 42
- (c) 102
- (d) Undefined behaviour
- (e) None of the above

d

8. Which of the following statements is **TRUE** regarding function prototypes in C? (2 marks)

- (a) If a function prototype is not declared before a function call, the C compiler will always generate a compilation error.
- (b) A function prototype allows the compiler to perform type checking on the arguments passed to the function during compilation.
- (c) Function prototypes are only required when the function is defined in a different source file.
- (d) A function prototype must include parameter names.
- (e) A function prototype must appear after the function definition in the same source file.

b

9. What is the output of the following C program (2 marks)

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i = 10;
6     while(i-->0)
7         if (i % 3 == 0)
8             printf("%d ", i);
9 }
```

- (a) 10 9 8 7 6 5 4 3 2 1
- (b) 9 8 7 6 5 4 3 2 1 0
- (c) 9 6 3 0
- (d) 8 5 2
- (e) The program will not compile due to syntax error on line 6

c

10. Consider the following C program:

(2 marks)

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int i = 5;
6     i = ++i + i++;
7     printf("%d ", i);
9 }
```

What is the output of the program?

- (a) 10
- (b) 11
- (c) 12
- (d) The program will not compile due to syntax error on line 6.
- (e) The program will compile but will have undefined behavior.

e

SECTION C Macros and Arrays

11. Consider the following C program:

(2 marks)

```
1 #include <stdio.h>
2
3 #define COUNT 1
4
5 int main(void)
6 {
7     int count = COUNT++;
8     printf("%d", count);
9     return 0;
10 }
```

What is the output of the program?

- (a) 1
- (b) 2
- (c) 3
- (d) The program will not compile due to syntax error on line 7.
- (e) The program will compile but will have undefined behavior.

d

12. What is the output of the following program?

(2 marks)

```
#include <stdio.h>

#define twice(x) x+x
#define thrice(y) (y+y)+(y)

int main(void)
{
    int x = 36/thrice(twice(6));
    printf("%d", x);
    return 0;
}
```

- (a) 13
- (b) 1
- (c) 36
- (d) 0
- (e) 27

a

13. Which of the following statements is **TRUE** regarding arrays in C? **(2 marks)**

- (a) If fewer initializers than the array size are provided, the remaining elements are automatically initialized to zero.
- (b) The size of an array can be changed after declaration using the assignment operator.
- (c) An array can be assigned a new initializer list after it has been declared.
- (d) If an array is partially initialized, the remaining elements contain random values.
- (e) The size of an array is always determined at runtime.

a

14. Consider the following function: **(2 marks)**

```
void foo(int arr[])
{
    printf("%d", sizeof(arr));
}
```

If the function is called as follows:

```
int data[10];
foo(data);
```

Which statement is **TRUE**?

- (a) The program will display 10.
- (b) The program will display the size of the entire array.
- (c) The program will display the size of a pointer.
- (d) The program will not compile.
- (e) None of the above.

c

15. Consider the following declaration: **(2 marks)**

```
int matrix[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

What is the value of `matrix[2][1]`?

- 1. 4
- 2. 5
- 3. 7
- 4. 8
- 5. None of the above

e

SECTION D Strings

16. Which of the following are valid declarations?

(2 marks)

- (i) `char *s1 = "Hello";`
- (ii) `char *s2 = "Hello" ", " "world";`
- (iii) `char *s3 = "Hello
world";`
- (iv) `char *s4 = "Hello \
world";`
- (v) `char *s5 = 'Hello';`

- (a) (i) and (ii) only
- (b) (i), (ii), and (iv) only
- (c) (i), (ii), (iii), and (iv) only
- (d) (ii), (iii), and (v) only
- (e) (i), (ii), (iv), and (v) only

b

17. What is the output of the following code fragment?

(2 marks)

```
char str[] = "One\0Two";  
printf("%s", str);
```

- 1. OneTwo
- 2. One
- 3. Two
- 4. No output is printed
- 5. Undefined behaviour

b

18. Given the following program:

(2 marks)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char p[20];
    char *s = "hello123";
    int len = strlen(s);
    int i;

    for (i = 0; i < len; i++)
        p[i] = s[len - i - 1];
    p[i] = '\\0';

    printf("%s",p);
    return 0;
}
```

What does the program print?

- (a) hello123
- (b) 123hello
- (c) 321olleh
- (d) No output is printed
- (e) Undefined behaviour

c

19. Consider the following code fragment:

(2 marks)

```
char str1[] = "Hello, world";
char *str2 = "Hello, world";
```

Determine whether the following statements are valid:

- (i) str1[0] = 'h';
- (ii) str2[0] = 'h';
- (iii) strcpy(str1, "NWEN241");
- (iv) strcpy(str2, "NWEN241");

- (a) (i) and (ii) only
- (b) (i) and (iii) only
- (c) (i), (ii) and (iii) only
- (d) (ii) and (iv) only
- (e) (ii), (iii) and (iv) only

b

SECTION E Structures

20. Consider the following declaration: (2 marks)

```
struct {
    int x;
    int y;
} point;
```

Which statement is **TRUE**?

- (a) The structure must have a tag to compile
- (b) The structure automatically becomes a typedef
- (c) The structure has no name but a variable is declared
- (d) The declaration `struct point p;` is a valid structure variable declaration
- (e) All of the above

c

21. Consider the following structure definition: (2 marks)

```
struct book {
    char title[32];
    char author[20];
    int year;
};
```

Which of the following is a **VALID** declaration in C, declaring a struct variable and initializing it at the same time?

- (a) `book b = {"The Hobbit", "J.R.R. Tolkien", 1937};`
- (b) `struct book b = ("The Hobbit", "J.R.R. Tolkien", 1937);`
- (c) `struct book b = {"The Hobbit", "J.R.R. Tolkien", 1937};`
- (d) `struct book b = {"The Hobbit"; "J.R.R. Tolkien"; 1937};`
- (e) `struct book = {"The Hobbit", "J.R.R. Tolkien", 1937};`

c

22. What is actually passed if you pass a structure variable to a function? (2 marks)

- (a) Copy of reference to structure variable.
- (b) Copy of starting address of structure variable.
- (c) Copy of ending address of structure variable.
- (d) Copy of structure variable.
- (e) None of the above.

d

23. Consider the following declarations:

(2 marks)

```
struct A {  
    int x;  
    int y;  
};
```

```
struct B {  
    struct A a;  
    int y;  
};
```

```
struct B b = {{3}, 5};
```

What is the value of `b.a.y`?

- (a) 0
- (b) 3
- (c) 5
- (d) Garbage value
- (e) Compilation error

a

SECTION F Pointers

24. Which of the following statements about pointers in C is **TRUE**? **(2 marks)**

- (a) A pointer variable always stores the value of another variable.
- (b) A pointer must always point to a valid memory location.
- (c) A pointer variable stores the memory address of another variable.
- (d) The size of a pointer depends on the data type it points to.
- (e) A pointer cannot point to another pointer.

c

25. Consider the following code fragment: **(2 marks)**

```
int x = 10;
int *p = &x;
*p = 20;
(*p)++;
```

What is the value of x after execution?

- (a) 10
- (b) 11
- (c) 20
- (d) 21
- (e) None of the above

d

26. Consider the following declarations: **(2 marks)**

```
int x = 10;
int *p = &x;
int **q = &p;
```

What does the expression ****q** represent?

- (a) The address of x
- (b) The value of x
- (c) The address of p
- (d) The value stored in p
- (e) None of the above

b

27. What is the output of the following program?

(2 marks)

```
#include <stdio.h>

int main(void)
{
    char *msg = "The quick brown fox";
    char *p = msg + 4;
    printf("%s", p);
    return 0;
}
```

- (a) The quick brown fox
- (b) quick brown fox
- (c) No output is printed
- (d) Compilation error
- (e) None of the above

b

For Questions 28–30:

Given the following variable declarations:

```
int a[] = {1,2,4,8,16};
int *ip = a;
int **pp = &ip;
```

Suppose that an int occupies 4 bytes in memory. The array a is at address 1000, while ip is at address 1100 and pp is at address 1200 (all addresses are in decimal).

28. What is the numeric value of the expression ip+1?

(2 marks)

- (a) 2
- (b) 1004
- (c) 1104
- (d) 1204
- (e) None of the above

b

29. What is the numeric value of the expression **pp+2?

(2 marks)

- (a) 1202
- (b) 1208
- (c) 3
- (d) 4
- (e) None of the above

c

30. What is the numeric value of the expression $*(*pp+*ip)$?

(2 marks)

- (a) 1
- (b) 2
- (c) 4
- (d) 1104
- (e) None of the above

b

SECTION G Storage Classes and Process Layout

31. In the following declaration: **(2 marks)**

```
register int i;
```

Which of the following is **TRUE** ?

- (a) The value of variable `i` is guaranteed to be stored in a CPU register.
- (b) If registers are all allocated, the compiler will store `i` in cache memory.
- (c) A register variable is local to the block which contains it.
- (d) The compiler can ignore the request, in which case the storage class defaults to `static`.
- (e) None of the above.

c

32. Consider the following C program: **(2 marks)**

```
#include <stdio.h>

void change() {
    extern int x;
    x = x + 2;
}

int x = 5;

int main(void)
{
    change();
    printf("%d\n", x);
    return 0;
}
```

What is the output of the program?

- (a) 5
- (b) 7
- (c) Compilation error
- (d) Undefined behaviour
- (e) None of the above

b

33. Which C storage class variables are stored in the **DATA** segment? **(2 marks)**

- (a) extern only
- (b) static only
- (c) register only
- (d) extern and static
- (e) static and register

d

34. Consider the following C program: **(2 marks)**

```
void foo(int a)
{
    int b;
    {
        int c;
    }
}

void bar(void)
{
    static int d;
}

int main(void)
{
    int e;
    foo(e);
    bar();
}
```

What will be the sequence of allocation and deletion of variables in the above code?

- (a) Allocate a, b, c, d, e ; Deallocate e, d, c, b, a;
- (b) Allocate d, e, a, b, c ; Deallocate c, b, a, e, d;
- (c) Allocate e, d, a, b, c ; Deallocate c, b, a, d, e;
- (d) Allocate e, a, b, c ; Deallocate c, b, a, e;
- (e) None of the above

b

35. Consider the following C program:

(2 marks)

```
#include <stdio.h>

void enigma(void)
{
    static int x = 2;
    x += 3;
    printf("%d ", x);
}

int main(void)
{
    enigma();
    enigma();
    static int x = 10;
    enigma();
    printf("%d ", x);
    enigma();

    return 0;
}
```

- (a) 5 8 11 10 14
- (b) 5 8 5 10 8
- (c) 5 8 11 14 10
- (d) 5 8 11 10 17
- (e) None of the above.

a

SECTION H Dynamic Memory Management

36. Why does the C programming language provide mechanisms for dynamic memory allocation? **(2 marks)**

- (a) To allow programs to allocate memory whose size can be determined during program execution, rather than fixed at compile time.
- (b) To allow programs to store variables permanently in the stack segment.
- (c) To ensure that all variables are automatically deallocated when a function returns.
- (d) To make all variables globally accessible across different source files.
- (e) None of the above.

a

37. In a C program, dynamically allocated memory is stored in which region of memory? **(2 marks)**

- (a) Data segment
- (b) Stack segment
- (c) Heap segment
- (d) Code segment
- (e) None of the above.

c

38. Which statement about `malloc()` and `calloc()` is **TRUE**? **(2 marks)**

- (a) Both initialize memory to zero
- (b) `malloc()` initializes memory to zero, `calloc()` does not
- (c) `calloc()` initializes memory to zero, `malloc()` does not
- (d) Neither initializes memory to zero
- (e) None of the above

c

39. Consider the following code snippet. Assuming the allocation is successful, the size (in bytes) of the memory block pointed to by `cp` will be: **(2 marks)**

```
char *cp;  
cp = calloc(40, sizeof(char));
```

- (a) 10 bytes
- (b) 20 bytes
- (c) 40 bytes
- (d) 80 bytes
- (e) None of the above

c

40. Consider the following code snippet: (2 marks)

```
char *ptr = (char *)malloc(8*sizeof(char));
realloc(ptr, 12*sizeof(char));
```

Suppose the call to `realloc()` on the second line is successful. Which of the following statements is **TRUE**?

- (a) The memory block pointed to by `ptr` is automatically expanded to 12 bytes, and `ptr` is guaranteed to point to the new memory location.
- (b) The program has a memory leak, because the return value of `realloc()` is not assigned to a pointer.
- (c) The call to `realloc()` always extends the existing block in place, so no new memory location is ever used.
- (d) The pointer `ptr` now automatically points to the last element of the resized memory block.
- (e) The call to `realloc()` frees the original memory block and sets `ptr` to `NULL`.

b

41. Consider the following C program: (2 marks)

```
#include <stdio.h>

int main(void)
{
    int *ip;
    ip = calloc(5, sizeof(int));
    if (ip != NULL) {
        for (int i=0; i<5; i++) {
            *ip = i;
            ip++;
        }
    }
    free(ip);
    return 0;
}
```

Which of the following is **TRUE** regarding the program?

- (a) The program correctly allocates memory, initializes the array, and safely frees the allocated memory.
- (b) The program results in a compilation error because pointer arithmetic cannot be performed on dynamically allocated memory.
- (c) The program will always crash because `free()` is called when the memory allocation fails.
- (d) The program causes undefined behaviour because `free()` is called on a pointer that no longer points to the start of the allocated memory.
- (e) None of the above

d

SECTION I User-Defined Types

42. Consider the following C code snippet: **(2 marks)**

```
enum { apple, banana, cherry = 5, date } myfruit = date;
```

What is the value of myfruit?

- (a) 3
- (b) 4
- (c) 5
- (d) 6
- (e) None of the above

d

43. Consider the following C program: **(2 marks)**

```
#include <stdio.h>

enum status { OFF, ON, ERROR };

int main(void)
{
    enum status light_status = OFF;
    printf("%d\n", light_status);
    return 0;
}
```

What is the main advantage of using enum status in this program?

- (a) It prevents light_status from being assigned values outside OFF, ON, and ERROR.
- (b) It improves code readability by giving meaningful names to integer constants.
- (c) It ensures light_status uses less memory than an int.
- (d) It automatically checks for invalid values at runtime.
- (e) All of the above

b

44. Consider the following code snippet:

(2 marks)

```
union U {
    char c;
    short s;
    long l;
} var;
```

What is the size of the variable var be equal to?

- (a) sizeof(char)
- (b) sizeof(short)
- (c) sizeof(long)
- (d) sizeof(char) + sizeof(short) + sizeof(long)
- (e) None of the above

c

45. Consider the following C program:

(2 marks)

```
#include <stdio.h>

union U {
    struct {
        int a;
        int b;
    } s1;
    struct {
        int x;
        int y;
    } s2;
};

int main(void)
{
    union U u;
    u.s1.a = 10;
    u.s1.b = 20;
    printf("%d %d\n", u.s2.x, u.s2.y);
    return 0;
}
```

Which of the following statements is **TRUE**?

- (a) The program will not compile due to syntax error.
- (b) The program will output 10 20.
- (c) The program will output 20 10.
- (d) The program will cause undefined behavior because different struct members are used in the printf() statement.
- (e) None of the above

b

SECTION J FILE Stream I/O

46. Which of the following sets of file streams are automatically opened and available to a C program when it begins execution? **(2 marks)**

- (a) input, output, error
- (b) stdin, stdout, stderr
- (c) stdin, stdout, stderr
- (d) fsin, fsout, fserr
- (e) None of the above

c

47. Which of the following will read a character from keyboard and store it in a character variable c? **(2 marks)**

- (a) gets(c);
- (b) c = getc();
- (c) gets(&c);
- (d) getchar(&c);
- (e) c = getchar();

e

48. Consider the following C code snippet: **(2 marks)**

```
int i;
FILE *fp = fopen("input.txt", 'r');
fscanf(fp, "%d", &i);
/* Done reading from the file */
printf("%d", i);
```

Which of the following describes an issue (error and poor programming) with the code?

- (a) The mode argument of fopen() should be a string, so 'r' should be replaced by "r".
- (b) After the call to fopen(), its return value must be checked to ensure that the file opening was successful.
- (c) After the call to fscanf(), its return value must be checked to ensure that the reading was successful.
- (d) The file must be closed using fclose().
- (e) All of the above.

e

49. Consider the following C code snippet:

(2 marks)

```
int c;
FILE *infp = fopen("infile.txt", "r");
FILE *outfp = fopen("outfile.txt", "w");
while( (c=getc(infp)) != EOF ) {
    fputc(c+1, outfp);
}
fclose(infp);
fclose(outfp);
```

Suppose `infile.txt` contains the following text:

Secret

What would be the contents of `outfile.txt` ?

- (a) Rdbqds
- (b) Secret
- (c) Tfdsfu
- (d) Empty file.
- (e) The numeric value of Secret + 1.

c

50. Which of the following statements is **TRUE**?

- (a) One advantage of file stream over file descriptor is that file descriptor allows content to be formatted using format specifiers.
- (b) To be able to read keyboard input, a program must first open the `stdin` stream.
- (c) When a file is opened with mode "a", the contents of the file (if it exists) will be deleted.
- (d) The function `fscanf()` will return EOF if the end of file is reached, or errors were encountered while reading the file.
- (e) None of the above

d

C Operator Precedence and Associativity

This page lists all C operators in order of their precedence (highest to lowest). Their associativity indicates in what order operators of equal precedence in an expression are applied.

Operator	Description	Associativity
() [] . ->	Parentheses (grouping) Brackets (array subscript) Member selection via object name Member selection via pointer	left-to-right
++ -- + - ! ~ (<i>type</i>) * & sizeof	Unary preincrement/predecrement Unary plus/minus Unary logical negation/bitwise complement Unary cast (change <i>type</i>) Dereference Address Determine size in bytes	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
	Logical OR	left-to-right
?:	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= = <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.