

Question Number - 1

Max Marks - 2

Answer/Marking Scheme:

- **Part a) (Row-major order):**

Use the formula:

$$\text{Address} = \text{Base} + [(i \times \text{number_of_columns}) + j] \times \text{size}$$

Here, $i = 3$, $j = 4$, $\text{number_of_columns} = 8$ and (assuming an integer occupies 4 bytes):

$$\text{Offset} = (3 \times 8 + 4) = 28$$

$$\text{Memory address} = 1600 + (28 \times 4) = 1600 + 112 = \mathbf{1712}$$

(Award 1 mark for the correct formula & calculation.)

- **Part b) (Column-major order):**

Use the formula:

$$\text{Address} = \text{Base} + [(j \times \text{number_of_rows}) + i] \times \text{size}$$

Here, $\text{number_of_rows} = 6$, so:

$$\text{Offset} = (4 \times 6 + 3) = 27$$

$$\text{Memory address} = 1600 + (27 \times 4) = 1600 + 108 = \mathbf{1708}$$

(Award 1 mark for the correct reasoning and answer.)

Question Number - 2

Max Marks - 18

Answer/Marking Scheme:

This question has several code snippets. Award marks for correct simulation of each snippet.

Part a) (First Code Snippet – 2 marks):

```
#include <stdio.h>

int main() {

    float fl;

    int i=40, j=30, k=20;

    int p=5;

    fl = 42/4 + 4.0/3 + 5.24;

    p = i > j > k;

    printf("fl= %.2f p=%d", fl, p);

}
```

Ideal Answer:

42/4 performs integer division → 10

4.0/3 ≈ 1.33 and 5.24 as given

fl = 10 + 1.33 + 5.24 = 16.57 (printed with two decimals)

For p: (i > j) is true (1), then (1 > k) → (1 > 20) is false → 0

Output:

f1= 16.57 p=0

(Award 1 mark each for correct f1 and p value.)

Part b) (Second Code Snippet – 2.5 marks):

```
#include <stdio.h>

void main() {

    char arr[] = {'l', 'a', 't', 'e', 's', 't'};

    char *p = (arr + 2);

    printf("%c", *p + 2);

    printf("\n%d %d", sizeof(arr), sizeof(p));

}
```

Ideal Answer:

Since `arr[2] = 't'`, then `*p = 't'` and `*t + 2` equals the character with ASCII value $(116 + 2) = 118$, which is 'v'.

`sizeof(arr)` gives 6 (6 characters \times 1 byte each).

`sizeof(p)` gives the size of a pointer (assume 4 bytes on a 32-bit system).

Output:

v

6 4 (or 8)

(Award 2.5 marks for identifying both outputs correctly. 1 mark if only 1 output is correct. 2 marks if 2 values are correct.)

Part c) (Third Code Snippet – 2.5 marks):

```
#include <stdio.h>

void main() {

    for (int k = 1; k < 4; )

        printf("%d \n", ++k);

}
```

Ideal Answer:

k starts at 1; then pre-increment (++k) prints:

Iteration 1: k becomes 2 → prints 2

Iteration 2: k becomes 3 → prints 3

Iteration 3: k becomes 4 → prints 4

Output:

2

3

4

(Award 2.5 marks for completely correct output.)

Part d) (Fourth Code Snippet – 3 marks):

```
#include <stdio.h>

int main() {

    int i = 0;

    for (i = 1; i < 20; i++) {

        switch(i) {

            case 1:

                i += 1;

            case 2:

                i += 3;

            case 4:

                i += 4;

            default:

                i += 8;

                break;

        }

        printf(" %d ", i);

    }

    return 0;

}
```

Ideal Answer:

Iteration when i = 1:

case 1: i becomes $1 + 1 = 2$
falls through case 2: i becomes $2 + 3 = 5$
falls through case 4: i becomes $5 + 4 = 9$
default: i becomes $9 + 8 = 17$
(Then $i++$ in loop makes $i = 18$) \rightarrow prints 17

Iteration when i = 18:

No case matches; default adds 8 \rightarrow i becomes $18 + 8 = 26$
(Then $i++$ in loop makes $i = 27$, which stops the loop) \rightarrow prints 26

Output:

17 26

(Award 1.5 marks for each of 17 and 26.)

Part e) (Fifth Code Snippet – 2 marks):

```
#include <stdio.h>

#define ALPHA 0

#define BETA 1

int main() {

    int i = 5;

    switch(i & 1) {

        default: printf("Default");

        case ALPHA: printf("alpha");

        case BETA: printf("beta");

    }

    return 0;

}
```

Ideal Answer:

i = 5; 5 & 1 equals 1.

The matching case is `case BETA:` (since BETA is defined as 1).

Execution starts at `case BETA:` and prints "beta".

(Award 2 marks for correctly identifying the jump to case BETA.)

Part f) (Sixth Code Snippet – 3 marks):

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

```
int main(){
```

```
    int k, sum = 0;
```

```
    for (k = 2048; k; k = k >> 1)
```

```
        sum++;
```

```
        printf("%d %o %x ", sum, sum + 1, sum + 2);
```

```
    return 0;
```

```
}
```

Ideal Answer:

The loop runs while k is nonzero. Since $2048 = 2^{11}$, it takes 12 shifts for k to become 0. Thus, sum = 12.

sum + 1 = 13, printed in octal \rightarrow 15 (since $13_{10} = 15_8$).

sum + 2 = 14, printed in hexadecimal \rightarrow e (since $14_{10} = e_{16}$).

Output:

12 15 e

(Award 1 mark for each of 12, 15 and e.)

Part g) (Seventh Code Snippet – 3 marks):

```
#include <stdio.h>

void main() {

    int i = 1, j = 5, k = 11;

    int *p = &j;

    int *q = p;

    int *r = &k;

    *p = i;

    (*p)++;

    i += 2;

    *r = *r - *q;

    p = r;

    j = j + i;

    k = k + *q;

    printf("%d %d %d ", i, j, k);

}
```

Ideal Answer:

Initially: i = 1, j = 5, k = 11

**p = i; sets j = 1*

*(*p)++; increments j to 2*

i += 2; sets i = 3

**r = *r - *q; computes $k = 11 - (\text{value pointed by } q = j = 2) \rightarrow k = 9$*

p = r; now p points to k

j = j + i; updates $j = 2 + 3 = 5$

*k = k + *q; uses *q (still j, which is 5) $\rightarrow k$ becomes $9 + 5 = 14$*

Output:

3 5 14

(Award 3 marks for correct pointer manipulation and output.)

Question Number - 3

Max Marks - 2

Answer/Marking Scheme:

The code for reversing an integer array is given with a mistake.

```
void reverse(int A[], int n) {  
  
    int i, j, temp;  
  
    i = 0;  
  
    while(i < n){  
  
        j = n-1-i;  
  
        temp = A[i];  
  
        A[i] = A[j];  
  
        A[j] = temp;  
  
        i++;  
  
    }  
  
}
```

Ideal Answer:

Encircle while($i < n$)

Correct Statement should be while($i < n/2$)

(Award 1 marks for each of above two lines)

Question Number - 4

Max Marks - 3

Answer/Marking Scheme:

Complete the code to transpose a square matrix in place (without using any additional array or new variable):

Ideal Answer:

Insert the following nested loop (after reading the matrix and before printing the transpose):

```
for( $i = 0; i < N; i++$ ) {  
  
    for( $j = i + 1; j < N; j++$ ) {  
  
        // Swap  $A[i][j]$  with  $A[j][i]$  without using an extra variable:  
  
         $A[i][j] = A[i][j] + A[j][i];$   
  
         $A[j][i] = A[i][j] - A[j][i];$   
  
         $A[i][j] = A[i][j] - A[j][i];$   
  
    }  
  
}
```

(Award 1 mark for setting correct loop bounds and 2 marks for correct in-place swapping technique.)

Question Number - 5

Max Marks - 5

Answer/Marking Scheme:

Examine the code for printing the multiplication table and correct the errors:

Error and Corrections:

Error 1: Closing braces for main() function is not present.

Error 2: In `scanf`, the variable is passed without an address operator.

Correction:

`scanf("%d", &n);`

Error 3: The while loop does not update (increment) the variable `factor`.

Correction:

Insert `factor++`; at the end of the loop's body.

(Optional) Add a newline in the printf inside the loop for better formatting.

(Award 1 mark for each correctly identified error and its correction; total 3 marks if three main errors are expected. Give 0.5 marks for optional statement if possible.)

Question Number - 6

Max Marks - 5

Answer/Marking Scheme:

Complete the C program that processes an input string as follows:

- **Task 1: Count the total number of characters that appear two or more times in the input string.**
- **Task 2: Remove all digits from the string.**
- **Task 3: Convert all alphabetic characters to lowercase.**
- **Task 4: Print the count and the modified string.**

Ideal Answer (Pseudo-code/Outline):

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

```
#define SZ 1000
```

```
void main() {
```

```
    char inp[SZ];
```

```
    int freq[128] = {0}; // Frequency table for ASCII characters
```

```
    int i, j = 0, repeatCount = 0;
```

```
    // Read the input string
```

```
    scanf("%s", inp);
```

```
    // Count frequency of each character in the input string
```

```
    for(i = 0; inp[i] != '\0'; i++){
```

```
        freq[(int)inp[i]]++;
```

```
}
```

```
// Count distinct characters that appear two or more times
```

```
for(i = 0; i < 128; i++){
```

```
if(freq[i] >= 2)
```

```
repeatCount++;
```

```
}
```

```
// Process the string: remove digits and convert alphabets to lowercase
```

```
for(i = 0; inp[i] != '\0'; i++){
```

```
// Skip digits
```

```
if(inp[i] >= '0' && inp[i] <= '9')
```

```
continue;
```

```
// Convert uppercase letters to lowercase manually
```

```
if(inp[i] >= 'A' && inp[i] <= 'Z')
```

```
inp[j++] = inp[i] + ('a' - 'A');
```

```
else
```

```
inp[j++] = inp[i];
```

```
}
```

```
inp[j] = '\0'; // Terminate the modified string
```

```

        // Print the required outputs

        printf("No. of characters that repeat = %d\n", repeatCount);

        printf("Output String: %s", inp);

    }

```

(Award 1.5 marks each for counting repeats, removing digits and changing to lowercase, and 0.5 marks for correctly printing the output string.)

Question Number - 7

Max Marks - 7

Answer/Marking Scheme:

*Write the code to delete the first node in a singly linked list whose data matches a given key. Use the NODE structure provided and the global pointer **head**.*

Ideal Answer (Pseudo-code/Outline):

```

void find_delete(int key) {

    NODE *temp = head, *prev = NULL;

    // If list is empty

    if(head == NULL)

        return;

    // If head node itself holds the key

```

```
if(head->data == key) {
```

```
temp = head;
```

```
head = head->next;
```

```
free(temp);
```

```
return;
```

```
}
```

```
// Traverse the list to find the key
```

```
while(temp != NULL && temp->data != key) {
```

```
prev = temp;
```

```
temp = temp->next;
```

```
}
```

```
// If key not found, no change is made
```

```
if(temp == NULL)
```

```
return;
```

```
// Delete the node and update links
```

```
prev->next = temp->next;
```

```
free(temp);
```

```
}
```

Marking Scheme:

- ***Handling deletion of head node: 2 marks***
- ***Traversing the list correctly (using two pointers) and stopping at the first occurrence: 3 marks***
- ***Properly updating the links and freeing memory: 2 marks***