2 Arrays

2.1 Let A be a two-dimensional array declared as follows:
A : array [1 ... 10] [1 ... 15] of integer;
Assuming that each integer takes one memory location. The array is stored in row-major order and the first element of the array is stored at location 100, what is the address of the element A[i][j]?

(a) 15i + j + 84 (b) 15j + i + 84
(c) 10i + j + 89 (d) 10j + i + 89

[1998 : 2 M]

2.2 An n × n array v is defined as follows:
v[i, j] = i – j for all i, j, $1 \le i \le n$, $1 \le j \le n$
The sum of the elements of the array v is

(a) 0 (b) n – 1
(c) $n^2 - 3n + 2$ (d) $\frac{n^2(n+1)}{2}$

[2000 : 1 M]

2.3 Suppose you are given an array s[1...n] and a procedure reverse(s, i, j) which reverses the order of elements in between positions i and j (both inclusive). What does the following sequence do, where $1 \le k \le n$:

```
reverse (s, 1, k);
reverse (s, k + 1, n);
reverse (s, 1, n);
```

(a) Rotates s left by k positions
(b) Leaves s unchanged
(c) Reverses all elements of s
(d) None of the above

[2000 : 1 M]

2.4 A program P reads in 500 integers in the range (0, 100) representing the scores of 500 students. If prints the frequency of each score above 50, what would be the best way for P to store the frequencies?

(a) An array of 50 numbers
(b) An array of 100 numbers
(c) An array of 500 numbers
(d) A dynamically allocated array of 550 numbers

[2005 : 1 M]

2.5 Let a be an array containing n integers in increasing order. The following algorithm determines whether there are two distinct numbers in the array whose difference is a specified number S > 0.

```
i = 0; j = 1;
while (j < n) {
    if (E) j++;
    else if (a[j] – a[i] = = S) break;
    else i++;
}
if (j < n) printf("yes");
else printf ("no");
```

Choose the correct expression for E.

(a) a[j] – a[i] > S (b) a[j] – a[i] < S
(c) a[i] – a[j] < S (d) a[i] – a[j] > S

[2005 : 2 M]

2.6 Let a and b be two sorted arrays containing n integers each, in non-decreasing order. Let c be a sorted array containing 2n integers obtained by merging the two arrays a and b. Assuming the arrays are indexed starting from 0, consider the following four statements:

I. a[i] ≥ b[i] => c[2i] ≥ a[i]
II. a[i] ≥ b[i] => c[2i] ≥ b[i]
III. a[i] ≥ b[i] => c[2i] ≤ a[i]
IV. a[i] ≥ b[i] => c[2i] ≤ b[i]

Which of the following is TRUE

(a) Only I and II (b) Only I and IV
(c) Only II and III (d) Only III and IV

[2005 : 2 M]

2.7 Consider the following C function in which size is the number of elements in the array E:

```
int MyX(int *E, unsigned int size) {
    int Y = 0;
    int Z;
    int i, j, k;
    for(i = 0; i < size; i++)
        Y = Y + E[i];
    for(i = 0; i < size; i++)
        for(j = i; j < size; j++)
        {
            Z = 0;
            for(k = i; k <= j; k++)
                Z = Z + E[k];
            if (Z > Y)
                Y = Z;
        }
    return Y;
}
```

The value returned by the function **MyX** is the

(a) maximum possible sum of elements in any sub-array of array **E**.
(b) maximum element in any sub-array of array **E**.
(c) sum of the maximum elements in all possible sub-arrays of array **E**.
(d) the sum of all the elements in the array **E**.

[2014 (Set-1) : 2 M]

2.8 Let *A* be a square matrix of size *n* × *n*. Consider the following pseudocode. What is the expected output?

```
C = 100;
for i=1 to n do
    for j=1 to n do
    {
        Temp = A[i][j] + C;
        A[i][j] = A[j][i];
        A[j][i] = Temp – C;
    }
for i=1 to n do
    for j =1 to n do
        output (A[i][j]);
```

(a) The matrix *A* itself
(b) Transpose of the matrix *A*
(c) Adding 100 to the upper diagonal elements and subtracting 100 from lower diagonal elements of *A*
(d) None of the above

[2014 (Set-3) : 1 M]

2.9 What is the output of the following C code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory.

```
int main()
{   unsigned int x[4][3] =
    {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
    printf("%u,%u, %u", x+3, *(x+3),*(x+2)+3);
}
```

(a) 2036, 2036, 2036 (b) 2012, 4, 2204
(c) 2036, 10, 10 (d) 2012, 4, 6

[2015 (Set-1) : 2 M]

2.10 Consider the following two *C* code segments. *Y* and *X* are one and two dimensional arrays of size *n* and *n* × *n* respectively, where $2 \le n \le 10$. Assume that in both code segments, elements of *Y* are initialized to 0 and each element *X*[*i*][*j*] of array *X* is initialized to *i* + *j*. Further assume that when stored in main memory all elements of *X* are in same main memory page frame.

Code segment 1:

```
// initialize elements of Y to 0
// initialize elements X[i][j] of X to i+j
    for (i = 0; i < n; i++)
        Y[i] + = X[0][i];
```

Code segment 2:

```
// initialize elements of Y to 0
// initialize elements X[i][j] of X to i+j
    for (i = 0; i < n; i++)
        Y[i] + = X[i][0];
```

Which of the following statements is/are correct?

S1: Final contents of array Y will be same in both code segments.
S2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory.
S3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory.

(a) Only S2 is correct
(b) Only S3 is correct
(c) Only S1 and S2 are correct
(d) Only S1 and S3 are correct

[2015 (Set-3) : 2 M]

2.11 Suppose $c = \langle c[0], \ldots, c[k-1] \rangle$ is an array of length *k*, where all the entries are from the set {0, 1}. For any positive integers *a* and *n*, consider the following pseudocode.

```
DOSOMETHING (c, a, n)
z ← 1
for i ← 0 to k – 1
    do z ← z^2 mod n
    if c[i] = 1
        then z ← (z × a) mod n
return z
```

If *k* = 4, *c* = ⟨1, 0, 1, 1⟩, *a* = 2 and *n* = 8, then the output of DOSOMETHING (*c*, *a*, *n*) is 0.

[2015 (Set-3) : 2 M]

2.12 Consider the following C program:

```
#include <stdio.h>
int main() {
        int a[4][5] = {{1, 2, 3, 4, 5},
                      {6, 7, 8, 9, 10},
                      {11, 12, 13, 14, 15},
                      16, 17, 18, 19, 20}};
        printf("%d"\n,*(*(a+**a+2)+3));
        return(0);
}
```

The output of the program is 19.

[2020 : 1 M]

2.13 Consider the following C program:

```
#include <stdio.h>
void stringcopy(char *, char *);
int main( ) {
        char a[30] = "@#Hello World!";
        stringcopy(a, a + 2);
        printf("%s\n", a);
        return 0;
}
void stringcopy(char *s, char *t) {
        while(*t)
                *s++ = *t++;
}
```

Which ONE of the following will be the output of the program?

(a) @#Hello World! (b) Hello World!
(c) ello World! (d) Hello World!d!

[2025 (Set-2) : 1 M]

2.14 An array A of length n with distinct elements is said to be bitonic if there is an index $1 \le i \le n$ such that $A[1..\,i]$ is sorted in the non-decreasing order and $A[i+1..n]$ is sorted in the non-increasing order.

Which ONE of the following represents the best possible asymptotic bound for the worst-case number of comparisons by an algorithm that searches for an element in a bitonic array A?

(a) $\Theta(n)$ (b) $\Theta(1)$
(c) $\Theta(\log^2 n)$ (d) $\Theta(\log n)$

[2025 (Set-2) : 2 M]

■■■■

Answers Arrays

2.1	(a)	**2.2**	(a)	**2.3**	(a)	**2.4**	(a)	**2.5**	(b)	**2.6**	(c)	**2.7**	(a)	**2.8**	(a)	**2.9**	(a)
2.10	(c)	**2.11**	(0)	**2.12**	(19)	**2.13**	(d)	**2.14**	(d)								

Explanations Arrays

2.1 (a)

Let r be number of elements in a row.
Address of the element A[i][j]

= Base address + (i – 1)* r + (j – 1)
= 100 + (i – 1) × 15 + (j – 1)
= 100 + 15i – 15 + j – 1
= 100 + 15i + j – 16
= 84 + 5i + j

2.2 (a)

Array is

$$\begin{bmatrix} 0 & -1 & -2 \;\ldots & 1-n \\ 1 & 0 & -1 \;\ldots & \\ 2 & 1 & 0 & \\ 3 & 2 & 1 & \\ \vdots & \vdots & \vdots & \\ -(1-n) & -(2-n) & & 0 \end{bmatrix}$$

Sum = 0 + 1 + 2 + ... + (–1) + (–2) + ... = 0.

2.3 (a)

Effect of the given 3 reversals for any k is equivalent to left rotation of the array of size n by k.

Let, S[1 .... 7]

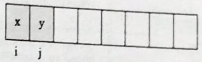
1	2	3	4	5	6	7

∴ n = 7, k = 2
reverse (S, 1, 2) we get [2, 1, 3, 4, 5, 6, 7]
reverse (S, 3, 7) we get [2, 1, 7, 6, 5, 4, 3]
reverse (S, 1, 7) we get [3, 4, 5, 6, 7, 1, 2]

2.4 (a)

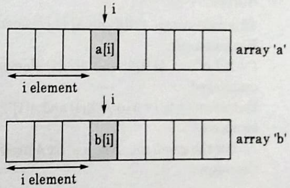
There are 500 students the score range is 0 to 100. Print the frequency of those student whose score above 50. So frequency range contains score from 50 to 100, so an array of 50 numbers is suitable for representing the frequency.

2.5 (b)



So, if condition will check a[j] – a[i] < S then increment j so that difference between two element increase and when a[j] – a[i] > S then increment i for making difference between two element indexed at i and j respectively is decrease. So, a[j] – a[i] < S is correct condition.

2.6 (c)



- When a[i] ≥ b[i] then 2i elements smaller than equal to a[i]. Since 'i' elements to array 'b' that are less than equal to b[i] and 'i' element of array 'a' which are less than equal to a[i]. So a[i] will comes after c[2i] i.e. c[2i] ≤ a[i] is true.
- When a[i] ≥ b[i] then b[i] always be less than equal to c[2i] i.e. when 'i' elements of array 'a' and 'i' elements of array 'b' are same then maximum index of b[i] is c[2i]. So, c[2i] ≥ b[i] is true.

2.7 **(a)**

```
for (i=0; i<size; i++)            → Sum of array stored in Y
Y=Y+E[i];
for (i=0; i<size; i++)            → Loop from 0 to (size-1)
{
  for (j=i; j<size; j++)          → Loop from i to (size-1)
  {
    Z=0;
    for (k=i; k<=j; k++)          → Computing sum of each
      Z=Z+E[K];                     subarray and stores in Z
    if (Z > Y)                    → If current sum of subarray is
    Y = Z;                          greater than previous then Y
  }                                 holds new maximum sum of
}                                   subarray value.
```

∴ The value (Y) returned by function is the maximum possible sum of elements in any subarray of array E.

2.8 **(a)**

Temp = $A[i][j] + C$;

$A[i][j] = A[j][i]$;

$A[j][i] = \text{Temp} - C$;

This code swaps $A[i][j]$ and $A[j][i]$ elements

For example

$i = 1, j = 2 \Rightarrow$ A[1][2] and A[2][1] elements are exchanged

But when $i = 2, j = 1 \Rightarrow A[2][1]$ and $A[1][2]$ again swapped

∴ For the given code, the matrix A itself will be the output.

2.9 **(a)**

0^{th} row	2000 1	2004 2	2008 3
1^{st} row	2012 4	2016 5	2020 6
2^{nd} row	2024 7	2028 8	2032 9
3^{rd} row	2036 10	2040 11	2044 12

Arithmatic of two-dimensional array addresses.

'x' is the base address of the 0^{th} row.

'x+1' is the base address of the 1^{st} row.

'x+i' is the base address of the i^{th} row.

Similarly *(x+i) is the i^{th} row. So *(x+i) the address of the 0^{th} element in the 0^{th} row (by putting i = 0)

The outputs printed by printf statements are:

x+3 → 2036 (starting address of 3^{rd} row)

*(x+3) → 2036 (address of 0^{th} element in 3^{rd} row)

*(x+2)+3 → 2036 (gives address in 2^{nd} row but +3 makes the starting address of 2^{nd} row (2024) incremented by $3 \times 4 = 12$)

∴ Output is 2036, 2036, 2036.

2.10 **(c)**

Y is one dimensional array

X is two dimensional array

Y	0	0	0	0	. . .	0
	0	1				$n-1$

$$X = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ \vdots \\ n-1 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & \cdots & n-1 \end{array} \\ \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & \cdots & n-1 \\ 1 & 2 & 3 & 4 & \cdots & n \\ 2 & 3 & 4 & 5 & \cdots & n+1 \\ 3 & 4 & 5 & 6 & \cdots & n+2 \\ \vdots & & & & & \\ n-1 & n & n+1 & n+2 & \cdots & n+n-2 \end{array}\right]_{n \times n} \end{array}$$

Code segment 1:

```
for (i = 0; i < n; i++)
    Y[i] + = X[0] [i];
```

Code segment 2:

```
for (i = 0; i < n; i++)
    Y[i] + = X[i] [0];
```

S1: Final contents of array Y will be same in both code segments.

Y	0	1	2	3	4	5	6	7		$n-1$
	0	1	2	3	4	5	6	7	. . .	$n-1$

is the output of both segments.

S2: X contents are stored row wise in C-program hence code segment row '0' accessed contiguous in memory.

S3: In code segment 2, content of X are not accessed in contiguous.

2.11 **(0)**

$K = 4$

0	1	2	3
1	0	1	1

$a = 2,\ n = 8,\ Z = 1,$

$\underline{i = 0}$ $Z = 1^2 \bmod 8 = 1$

if (C[0] = = 1) $Z = (1 \times 2) \bmod 8 = 2$

$\underline{i = 1}$ $Z = 2^2 \bmod 8 = 4$

$\underline{i = 2}$ $Z = 4^2 \bmod 8 = 0$

if (C[2] = = 1) $Z = (0 \times 2) \bmod 8 = 0$

$\underline{i = 3}$ $Z = 0^2 \bmod 8 = 0$

if (C[3] = = 1) $Z = (0 \times 2) \bmod 8 = 0$

Returns 0

2.12 **(19)**

'a' is a two dimensional array as shown below:

0, 0 1	0, 1 2	0, 2 3	0, 3 4	0, 4 5
1, 0 6	1, 1 7	1, 2 8	1, 3 9	1, 4 10
2, 0 11	2, 1 12	2, 2 13	2, 3 14	2, 4 15
3, 0 16	3, 1 17	3, 2 18	3, 3 19	3, 4 20

a = address of 0^{th} index 1-D array

*a = address of 0^{th} index element of 0^{th} index 1-D array

**a = value of 0^{th} index element of 0^{th} index 1-D array

**a = 1

**a + 2 = 1 + 2 = 3

a + 3 = address of 3^{rd} index 1-D array

*(a + 3) = address of 0^{th} index element of 3^{rd} index 1-D array

*(a + 3) + 3 = address of 3^{rd} index element of 3^{rd} index 1-D array

((a + 3) + 3) = value of 3^{rd} index element of 3^{rd} index 1-D array = 19

Hence it will print 19.

2.13 **(d)**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	@	#	H	e	l	l	o		w	o	r	l	d	!	/o
	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114

s: ~~100~~ ~~101~~ ~~102~~ ~~103~~ ~~104~~ ~~105~~ ~~106~~ ~~107~~ ~~108~~ ~~109~~ ~~110~~ ~~111~~ 112

t: ~~102~~ ~~103~~ ~~104~~ ~~105~~ ~~106~~ ~~107~~ ~~108~~ ~~109~~ ~~110~~ ~~111~~ ~~112~~ ~~113~~

- Value at location 102 (H) copy into value at location 100.
- Value at location 103 (e) copy into value at location 101.

⋮

- The same process continues until it reaches the null character.

2.14 **(d)**

	1	2	3	4	5	6	7	8	9	10
a	4	6	8	10	12	15	18	12	8	7

Using Binary Search:

```
low = 1; high = n;
while(low ≤ high)
{               m = (low + high)/2;
    if (a[mid - 1] < a[mid] < a[mid + 1])
        low = mid + 1;
    else if a[mid - 1] > a[mid] > a[mid + 1]
        high = mid - 1;
    else if (a[mid - 1] > a[mid] > a[mid + 1])
        return(mid);
}
```

TC: $\theta(\log n)$