Note: I have expressed the thinking process using red ink in a box. This will help you how to approach questions.

Q. 1:
(i) Convert 238 decimal to octal

\[
\begin{array}{c|c|c|c|c|c}
8 & 238 & 6 & 2 & 3 & 3 \\
8 & 29 & 6 & 3 & 3 & 2 \\
8 & 3 & 5 & 2 & 2 & 1 \\
8 & 0 & 3 & 1 & 1 & 0 \\
\end{array}
\]

\[ (238)_{10} = (356)_{8} \]

(ii) Convert A3D hexadecimal to decimal

\[
\begin{align*}
A_6 & \rightarrow (10)_{10} \quad \therefore (A3D)_{16} = 10 \times 16^2 + 3 \times 16^1 + A \times 16^0 \\
D_6 & \rightarrow (13)_{10} \\
\end{align*}
\]

\[ = 1600 + 48 + 13 \\
= (2621)_{10} \]

\[ (A3D)_{16} = (2621)_{10} \]

Q. 2:
(b) Write a C.P. that will convert a decimal number into any base.

**Thought Process:**

→ To convert any number 'num' into base 'base', we keep on dividing that number by base.
→ The remainder 'rem' we get forms the required number
→ We keep storing the remainder in an array and simply reverse it.

**Example:**

\[ (10)_{10} \rightarrow (?)_{6} \]

```
2 | 10 | 5 | 2 | 1 | 0
---+---+---+---+---+---
    |   |   |   |   |   
```

→ This reversing can be done using a simple for loop from i = 0 to i = count - 1
→ If remainder < 10, remainder stays as it is.
→ else if remainder ≥ 10, we have to store it in character form.

\[ 10 \rightarrow A \]

\[ 11 \rightarrow B \]

→ Our array storing remainder is integer. To store character remainders, we simply add 55 to the remainder.

\[ \%d \quad \%c \]

```
if rem = 10
   rem = rem + 55
   \%d \quad \%c

if rem = 11
   rem = rem + 55
   \%d \quad \%c
```

Now, have a look at the program and come back to thought process to understand any step.
Que 2.
(c) Write a program to calculate the sum of the following series without using pow() library function.

\[ S = 1 - \frac{x^1}{2!} - \frac{x^3}{3!} - \ldots \] n terms

Thought Process:

\[ S = 1 - \frac{x^1}{2!} - \frac{x^3}{3!} + \ldots \] n terms

\[ = \frac{x^0}{0!} - \frac{x^1}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots \] n terms

\[ = \frac{x^0}{0!} - \frac{x^1}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots \] n terms

\[ \therefore S = \sum_{i=0}^{n-1} (-x)^i \rightarrow n = \text{no of terms} \]

for loop

\[ \text{for loop will calculate numerator and denominator} \]

\[ \text{Calculate power using recursion} \quad \text{and} \]

\[ \text{Calculate factorial using a } j \text{ for loop} \]

\[ \text{as pow()} \text{ is not allowed} \]

\[ \text{The pow function is defined outside main()} \text{ and calculated numerator.} \]

\[ \text{Denominator is assigned to factorial output.} \]

\[ \text{Then final sum is calculated and printed.} \]

Que 3.

a. Write a C.P. to arrange the numbers stored in an array in such a way that the array will have the odd numbers followed by even numbers.
**Thinking process:**

→ This is a trivial nested loops problem as follows.

**Question 3.6:** Write a program that accepts the word from a user and prints it in the following way. For example, if the word is “STUDY”, the program will print it as:

```
S
ST
STU
STUD
STUDY
```

**Output:**

```
0 1 2 3 4
```

```c
#include <stdio.h>

int main()
```

```c
    char str[10];
    int i, j, rows = 0, columns = 0, n = 0, length = 0;

    printf("Enter the string: ");
    scanf("%s", str);

    printf("Input string: %s\n", str);

    length = strlen(str);

    rows = length;
    columns = 5;

    for (i = 0; i < rows; i++)
        for (j = 0; j < columns; j++)
            printf("%c", str[i]);
```

```c
    return 0;
```
Question 4.

a) Write a CP that will read a word and rewrite it in alphabetized order. Given: if the word is “maton”, the program should print “atnom”.

Thinking process:

- For every i, we compare it with all the elements ahead of it. 
  - j loop from i+1 to j=n-1
  - if any element is lower than i, we swap i with that element.

eq: $S: \text{ba}t_i \quad n=3$

Loop 1: $i=0 \quad S: \text{ba}t_i$

1. $i=0 \quad j=i+1=1$
   - if $i=0$ with $j=1$
     - as $b \geq a$, swap
     - $S: \text{a}b\text{t}_i$ $i=0 \quad j=2$

Now $i=0$ with $j=2$

2. $i=0 \quad j=i+1=1$
   - if $i=0$ with $j=1$
     - as $a \geq t$, no swap
     - $S: \text{ab}t_i$ $i=0 \quad j=2$

Loop 2: $i=1 \quad S: \text{ab}t_i$

1. $i=1 \quad j=i+1=2$
   - $j=2$ (t)
   - $i=1 \quad j=2$
   - as $b < t$, no swap

Letters sorted.
(b) Write a recursive C++ to find the factorial of a given number.

Thinking process:

\[ n! = n \times (n-1)! \]

\[ n! = n \times (n-1)! \]

Thus,

\[ \text{fact}(n) = n \times \text{fact}(n-1) \]

\[ \text{fact}(n-1) = (n-1) \times \text{fact}(n-2) \]

\[ \text{fact}(n-2) = (n-2) \times \text{fact}(n-3) \]

...\[ \text{fact}(2) = 2 \times \text{fact}(1) \]

\[ \text{fact}(1) = 1 \]

Use of recursion to reach the base case i.e., \( \text{fact}(1) = 1 \).


Use of recursion to reach the base case i.e., \( \text{fact}(1) = 1 \).

Then, using all calculated values to find \( \text{fact}(n) \) in the end.

\[ \text{fact}(4) = 4 \times \text{fact}(3) \]

\[ \text{fact}(3) = 3 \times \text{fact}(2) \]

\[ \text{fact}(2) = 2 \times \text{fact}(1) \]

\[ \text{fact}(1) = 1 \]

Thus, \( \text{fact}(4) = 24 \).

```cpp
#include<iostream>

int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}

int main()
{
    int n;
    std::cout << "Enter a positive integer: " << std::endl;
    std::cin >> n;
    std::cout << "Factorial of \( n \) is \( n! \) \n    " << fact(n) << std::endl;
    return 0;
}
```

(b) Declare a structure to store the information of 10 cricketers:

1. Cricketer name
2. Matches played
3. Runs scored
4. Strike Rate.

Use a function to display cricketer information having maximum strike rate.

Thinking process:

- Step 1: Create a structure with required information.
- Step 2: Accepting data from user
- Step 3: Finding maximum strike rate: Assume 1st element as max
- Step 4: Display player details w.r.t. maximum strike rate.

```cpp
struct cricketer
{
    char name[20];
    int matches, runs, strike;
    char name[20];
}

int main()
{
    int n = 10;
    cricketer player[10];
    for (int i = 0; i < n; i++)
    {
        std::cout << "Enter name, matches, runs, strike: " << std::endl;
        std::cin >> player[i].name >> player[i].matches >> player[i].runs >> player[i].strike;
    }
    int max_strike = player[0].strike;
    for (int i = 1; i < n; i++)
    {
        if (player[i].strike > max_strike)
        {
            max_strike = player[i].strike;
        }
    }
    std::cout << "Maximum strike rate: " << max_strike << std::endl;
    return 0;
}
```
Question 6:
(a) Write C code to add two pointers.

Thinking process:

```
int *p, *q;

p = &num1; → p stores address of num1
q = &num2; → q stores address of num2

Sum = *p + *q → value at address stored in p + value at address stored in q
     = 400 + 102
     = 502
```

```
Sum = 4 + 3 = 7
```