***PART-1***

**PROGRAMMING COMPUTER PROJECT**

**Part-1: The Division Algorithm of binary numbers**

1. Write a subroutine to subtract two binary numbers:

**procedure Subtraction** (a, b, d: integer)

for i := 0 to n-1 {Complement of negative binary number}

if bi == 0 then

bi = 1

else

bi = 0

c := 0 {add positive binary with negative binary numbers}

for j := 0 to n − 1

d := ⌊(aj + bj + c) ∕ 2⌋

sj := aj + bj + c − 2d

c := d

sn := c

return (s0, s1, …... sn)

**Main program:**

MAIN

BEGIN

Call Read (A); input A10

Call Read (B); input B10

Call Convert (A, a); convert the A10 to a2

Call Convert (B, b); convert the B10 to b2

Call Subtraction (a, b, q, r); ‘q’ = quotient and ‘r’= reminder

Call Print (q, r); Print ‘q’ and ‘r’

END

1. Write a subroutine to compute div and mod of two binary numbers:

**procedure Division** (a, b, q, r: integer)

q := 0

r := |a|

**while** r ≥ b

r := r − b

q := q + 1

**if** a < 0 and r > 0

**then**

r := b − r

q := −(q + 1)

**return** (q, r)

**{**Here ‘q’ is quotient, and ‘r’ is reminder}

**Main program:**

MAIN

BEGIN

Call Read (A); input A10

Call Read (B); input B10

Call Convert (A, a); convert the A10 to a2

Call Convert (B, b); convert the B10 to b2

Call Division (a, b, q, r); ‘q’ = quotient and ‘r’= reminder

Call Print (q, r); Print ‘q’ and ‘r’

END

**STEP NO.2**

1. **For k = 37:**

initially b=2, q=37, k =0

For 1st iteration, when k = 0:

While (q=37) ≠ 0

A0 = (37 mod 2) = 1

Q = (37 div 2) = 18

**K = k+1 = 1**

For 2nd iteration, when k = 1:

While (q=18) ≠ 0

A1 = (18 mod 2) = 0

Q = (18 div 2) = 9

**K = k+1 = 2**

For 3rd iteration, when k = 2:

While (q=9) ≠ 0

A2 = (9 mod 2) = 1

Q = (9 div 2) = 4

**K = k+1 = 3**

For 4th iteration, when k = 3:

While (q=4) ≠ 0

A3 = (4 mod 2) = 0

Q = (4 div 2) = 2

**K = k+1 = 4**

For 5th iteration, when k = 4:

While (q=2) ≠ 0

A4 = (2 mod 2) = 0

Q = (2 div 2) = 1

**K = k+1 = 5**

For 6th iteration, when k = 5:

While (q=1) ≠ 0

A5 = (1 mod 2) = 1

Q = (1 div 2) = 0

**K = k+1 = 6**

For 7th iteration, when k = 6:

While (q=0) ≠ 0

Now, as loop condition become false. So, loop will break;

Return

A5 A4 A3 A2 A1 A0 = 100101

Output will be:

**(37)10 = (100101)2**

1. **For k = 22:**

initially b=2, q=22, k =0

For the 1st iteration, when k = 0:

While (q=22) ≠ 0

A0 = (22 mod 2) = 0

Q = (22 div 2) = 11

**K = k+1 = 1**

For the 2nd iteration, when k = 1:

While (q=11) ≠ 0

A1 = (11 mod 2) = 1

Q = (11 div 2) = 5

**K = k+1 = 2**

For the 3rd iteration, when k = 2:

While (q=5) ≠ 0

A2 = (5 mod 2) = 1

Q = (5 div 2) = 2

**K = k+1 = 3**

For the 4th iteration, when k = 3:

While (q=2) ≠ 0

A3 = (2 mod 2) = 0

Q = (2 div 2) = 1

**K = k+1 = 4**

For the 5th iteration, when k = 4:

While (q=1) ≠ 0

A4 = (1 mod 2) = 1

Q = (1 div 2) = 0

**K = k+1 = 5**

For the 6th iteration, when k = 5:

While (q=0) ≠ 0

Now, as loop condition become false. So, loop will break;

Return

A4 A3 A2 A1 A0 = 10110

Output will be:

**(22)10 = (10110)2**

**(BINARY DIVISION)**

**From Step No.1**

**(37)10 = (100101)2**

**(22)10 = (010110)2**

Initially, q = 0, r = |a| = **100101**, b = **010110**

1st iteration, r = 100101, b = 010110:

While r >= b (true)

r = r – b

= 100101- 010110

= 01111

q = 0+1 = 01

2nd iteration, r = 01111, b = 010110:

While r >= b (false, so loop will break)

if a < 0 and r > 0 (**true**)

then

r = b – r

= 010110 – 01111

= 0111

q = - (q+1)

= -(01+1)

= -(10)

q = 01 (After complement)

**So,**

**Reminder = r = 01111**

**Quotient = q = 01**

**(BINARY SUBTRACTION)**

**From Step No.1**

**(37)10 = (100101)2**

**(22)10 = (010110)2**

For subtraction, we have to take 2’ complement of **(22)10 = (010110)2**

So,

= 010110

First complement is

= 101001

Add 1 to it:

= 101010

**So, 2’ complement of 22 = 101010**

**Now,** we just have to add binary of 37 and 2’ complement of 22.

(37)10 = (A5 A4 A3 A2 A1 A0)= (100101)2

(-22)10 = (B5 B4 B3 B2 B1 B0)= (101010)2

**Initially, c=0, j=0, n=6**

first iteration, j = 0:

For j=0 to 5

D = (A0=1 + B0=0 + C=0) / 2 = 0

S0 = A0=1 + B0=0 + C=0 – 2(0) = 1

C = 0

second iteration**, when j = 1:** For j=1 to 5

D = (A1=0 + B1=1 + C=0) / 2 = 0

S1 = A1=0 + B1=1 + C=0 – 2(0) = 1

C = 0

third iteration**, j = 2:**

For j=2 to 5

D = (A2=1 + B2=0 + C=0) / 2 = 0

S2 = A2=1 + B2=0 + C=0 – 2(0) = 1

C = 1

fourth iteration, j = 3:

For j=3 to 5

D = (A3=1 + B3=1 + C=1) / 2 = 1

S3 = A3=1 + B3=1 + C=1 – 2(1) = 1

C = 0

fifth iteration, j = 4:

For j=4 to 5

D = (A4=0 + B4=0 + C=0) / 2 = 0

S4 = A4=0 + B4=0 + C=0 – 2(0) = 0

C = 0

sixth iteration, j = 5:

For j=5 to 5

D = (A5=1 + B5=1 + C=0) / 2 = 1

S5 = A5=1 + B5=1+ C=0 – 2(0) = 0

C = 0

7th iteration, **j = 6:**

For loop breaks here.

So, it will return,

**S5S4S3S2S1S0 = 001111**

**= (22)10**

**Part II: Writing Project**

**3. Explain how sorting algorithms can be classified into a taxonomy based on the underlying principle on which they are based**

Sorting algorithm can be classified into 4 categories:

1. Number of Swaps or Inversion

e.g., selection sort.

1. Recursion or Non-Recursion

Recursive: Quick Sort, Merge Sort

Non-Recursion: Selection Sort, Insertion Sort

1. Stability Sorting algorithms

Insertion sort, Merge Sort, and Bubble Sort are stable

Heap Sort and Quick Sort are not stable

1. Extra Space Requirement

Insertion sort and Quick-sort takes extra space

Merge Sort does not take any extra space

**8. Develop a detailed list of algorithmic paradigms and provide examples using each of these paradigms.**

These are the few algorithmic paradigms:

1. Backtracking

e.g., eight queen puzzle, cross words, sudoku and knapsack problem.

1. Branch and bound

e.g., combinational, mathematical optimization and non-linear programming.

1. Brute-force search

e.g., cryptography

1. Divide and conquer

Mostly use in sorting algorithms (merge sort and quick sort)

1. Dynamic programming

e.g., Dijkstra's algorithm, Fibonacci sequence, and Checkerboard

1. Greedy algorithm

e.g., Kruskal's algorithm and Decision tree learning