

Exploring Computer Science Concepts

Via ACSL Competitions

Number Systems

- Decimal
 - base 10
- Binary
 - base 2
- Octal
 - base 8
- HexaDecimal
 - base 16

Digits per base

- Decimal
 - Ten digits
 - 0 1 2 3 4 5 6 7 8 9
- Binary
 - Two Digits
 - 0 1
- Octal
 - 8 digits
 - 0 1 2 3 4 5 6 7
- Hexadecimal
 - 16 digits
 - 0 1 2 3 4 5 6 7 8 9 A B C D E F

Counting in Decimal

- Increment the lowest place value (right most digit)
- When last digit is reached
 - Set the current column to 0
 - Increment the column on the left by 1
- Lets count with decimal
 - 0 to 10
 - 95 to 105
- How frequently do we add a new digit ?

Decimal	Decimal
0	95
1	96
2	97
3	98
4	99
5	100
6	101
7	102
8	103
9	104
10	105

Counting in other bases

- Counting in binary
 - We add a new digit frequently
 - At 2 , 4, 8, 16 ... decimal values
- Counting in Octal
 - We add a new digit at every
 - At 8, 64,128 ...
- Counting in HexaDecimal
 - We add a new digit at every
 - At 16, 256 ... values

Decimal	Binary	Octal	HexaDecimal
0	0	0	1
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Place Values

- Decimal

- *We deal in power of 10s*

- $2452 = 2 * 1000 + 4 * 100 + 5 * 10 + 2 * 1$

- $101 = 1 * 100 + 0 * 10 + 1 * 1$

- Binary

- *We deal in power of 2s*

- $111 = 1 * 4 + 1 * 2 + 1 * 1$

- $1010 = 1 * 8 + 0 * 4 + 1 * 2 + 0 * 1$

Place values

Converting binary → decimal

Decimal	Binary
<p>2452 →</p> $\begin{aligned} 2 \times 10^3 &= 2000 \\ 4 \times 10^2 &= 400 \\ 5 \times 10^1 &= 50 \\ 2 \times 10^0 &= 2 \\ &= 2452 \end{aligned}$	<p>111 →</p> $\begin{aligned} 1 \times 2^2 &= 1 \times 4 = 4 \\ 1 \times 2^1 &= 1 \times 2 = 2 \\ 1 \times 2^0 &= 1 \times 1 = 1 \\ &= 7 \end{aligned}$
<p>101 →</p> $\begin{aligned} 1 \times 10^2 &= 100 \\ 0 \times 10^1 &= 0 \\ 1 \times 10^0 &= 1 \\ &= 101 \end{aligned}$	<p>1010 →</p> $\begin{aligned} 1 \times 2^3 &= 1 \times 8 = 8 \\ 0 \times 2^2 &= 0 \times 4 = 0 \\ 1 \times 2^1 &= 0 \times 2 = 2 \\ 0 \times 2^0 &= 0 \times 1 = 0 \\ &= 10 \end{aligned}$
<p>756 →</p>	<p>1000 →</p>

Converting Binary to Decimal

128 64 32 16 8 4 2 1

$$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 = 64 + 16 + 1 = 81$$

$$1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 = 128 + 4 + 2 + 1 = 135$$

$$1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 = ?$$

Converting Decimal to Binary

128 64 32 16 8 4 2 1

= 22

= 33

= 130

Exercises

- Convert following Decimal numbers to binary
 - 127_{10}
 - 128_{10}
 - 129_{10}
 - 255_{10}
 - 256_{10}
- Convert following Binary numbers to Decimal
 - 101101_2
 - 1110_2
 - 1111_2
 - 0110_2

Challenge

- How does binary addition and subtraction work ?

Place values

Converting octal,hexadecimal→decimal

Octal → Decimal	HexaDecimal → Decimal
<p>777 →</p> $\begin{aligned}7 \times 8^2 &= 7 \times 64 = 448 \\7 \times 8^1 &= 7 \times 8 = 56 \\7 \times 8^0 &= 7 \times 1 = 7 \\&= 511\end{aligned}$	<p>2AB →</p> $\begin{aligned}2 \times 16^2 &= 2 \times 256 = 512 \\A \times 16^1 &= 10 \times 16 = 160 \\B \times 16^0 &= 11 \times 1 = 11 \\&= 683\end{aligned}$
<p>137 →</p> $\begin{aligned}1 \times 8^2 &= 1 \times 64 = 64 \\3 \times 8^1 &= 3 \times 8 = 24 \\7 \times 8^0 &= 7 \times 1 = 7 \\&= 95\end{aligned}$	<p>101 →</p> $\begin{aligned}1 \times 16^2 &= 1 \times 256 = 256 \\0 \times 16^1 &= 0 \times 16 = 0 \\1 \times 16^0 &= 1 \times 1 = 1 \\&= 257\end{aligned}$
<p>756 →</p>	<p>4A3 →</p>

Converting Decimal to Octal

$$(312)_{10} \rightarrow (?)_8$$

$$312 / 8 \rightarrow \text{quotient : } 39 \text{ , Remainder } 0$$

$$39 / 8 \rightarrow \text{quotient : } 4 \text{ , Remainder } 7$$

$$4 / 8 \rightarrow \text{quotient : } 0 \text{ , Remainder } 4$$

$$\mathbf{(470)_8}$$

$$(112)_{10} \rightarrow (?)_8$$

$$112 / 8 \rightarrow \text{quotient : } 14 \text{ , Remainder } 0$$

$$14 / 8 \rightarrow \text{quotient : } 1 \text{ , Remainder } 6$$

$$1 / 8 \rightarrow \text{quotient : } 0 \text{ , Remainder } 1$$

$$\mathbf{(160)_8}$$

Exercises

- Convert following Decimal numbers to Octal
 - 111_{10}
 - 88_{10}
 - 511_{10}
 - 512_{10}
 - 513_{10}
- Convert following Octal numbers to Decimal
 - 45_8
 - 77_8
 - 100_8
 - 101_8

Converting Decimal to Hexadecimal

$$(312)_{10} \rightarrow (?)_8$$

$$312 / 16 \rightarrow \text{quotient: } 19, \text{ Remainder } 8$$

$$19 / 16 \rightarrow \text{quotient: } 1, \text{ Remainder } 3$$

$$1 / 16 \rightarrow \text{quotient: } 0, \text{ Remainder } 1$$

$$\mathbf{(138)}_{16}$$

$$(112)_{10} \rightarrow (?)_8$$

$$112 / 16 \rightarrow \text{quotient: } 7, \text{ Remainder } 0$$

$$7 / 16 \rightarrow \text{quotient: } 0, \text{ Remainder } 7$$

$$\mathbf{(70)}_{16}$$

Hexadecimal → Octal

$(AC)_{16} \rightarrow (1010)_2 \times 16 + (1100)_2 \times 1$ // Replace hexa with binary
→ $(1010\ 1100)_2$ // Convert to binary
→ $(010\ 101\ 100)_2$ // group by 3s , added extra 0s in the front
→ $(254)_8$ // Replace each group by octal value

$(1EF)_{16} \rightarrow (0001)_2 \times 256 + (1110)_2 \times 16 + (1111) \times 1$
→ $(0001\ 1110\ 1111)_2$ // Convert to binary
→ $(\cancel{000}\ 111\ 101\ 111)_2$ // group by 3 , removed 0s in the front
→ $(757)_8$ // Replace each group by octal value

Octal \rightarrow Hexadecimal

- (**757**)₈ \rightarrow (?)₁₆
- (**254**)₈ \rightarrow (?)₁₆

Exercises

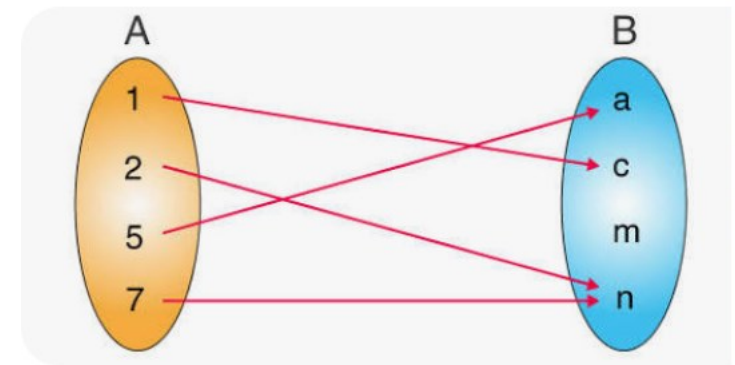
- Convert following Decimal numbers to HexaDecimal/Octal
 - 255_{10}
 - 256_{10}
 - 257_{10}
- Convert following HexaDecimal numbers to Decimal, Octal
 - 99_{16}
 - 100_{16}
 - 101_{16}

Recursion : programming

- Create factorial
 - $5! = 5 \times 4 \times 3 \times 2 \times 1 = \mathbf{120}$
 - $2! = \mathbf{2}$
- Provide sum of Fibonacci numbers using recursion
 - fibonacci(13)
 - $0 + 1 + 1 + 2 + 3 + 5 + 8 + 13 = \mathbf{33}$
 - fibonacci(3)
 - $0 + 1 + 1 + 2 + 3 = \mathbf{7}$

Relations

- A **relation** : connection/mapping between elements of two or more sets, Some characteristics ...
 - It is a **mapping** as *shown in figure*
 - Can be written as **Ordered pairs**
 - $\{(1,c), (2,n), (5,a), (7,n)\}$
 - Not always unique
 - E.g. $y^2 = 4$ has multiple solutions (how many?)
- Examples
 - Numerical relationship : $4+3 = 7$
 - Equation : $y = 2x+3$
 - Geometry : two congruent triangles
 - Set theory : A is a subset of B



Functions , mathematical kind

- A **relation** that gives exactly **one unique output for each input**

$x \rightarrow \boxed{\text{Rule}} \rightarrow y$:this is a function as you always get one answer

$x \rightarrow \boxed{\text{Rule}} \begin{matrix} \nearrow a \\ \rightarrow b \\ \searrow c \end{matrix}$:Not a function as you get multiple answers

- Representation
 - $f(x) = 2x+1$: this is a function
 - $g(x) = \pm 3x$: this is **not** a function , why?
- Follow up reading (optional): pg 11-13 : [functions](#)

Evaluating functions

- Solving
 - Substitute variables with numerals
 - Evaluate
 - Follow PEMDAS/BODMAS
- Solve for $x = 0, 1, 2, 3$
 - $f(x) = 3x + 1$
 - $g(x) = 2x^2 + 3$
 - $h(x) = x^2 + 2x + 1$

Recursive Functions

- Functions calling themselves

Fibonacci numbers	$fib: \mathbb{N} \rightarrow \mathbb{N}$ $fib(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ fib(n-1) + fib(n-2), & \text{if } n \geq 2. \end{cases}$
Factorial	fact(n) = n * fact(n-1) {given, $fact(1) = fact(0) = 1$ }
Lucas numbers (same rule as Fibonacci but with different starting values)	l (n) = l(n-1) + l(n-2) {given , $l(0) = 2, l(1) = 1$ }

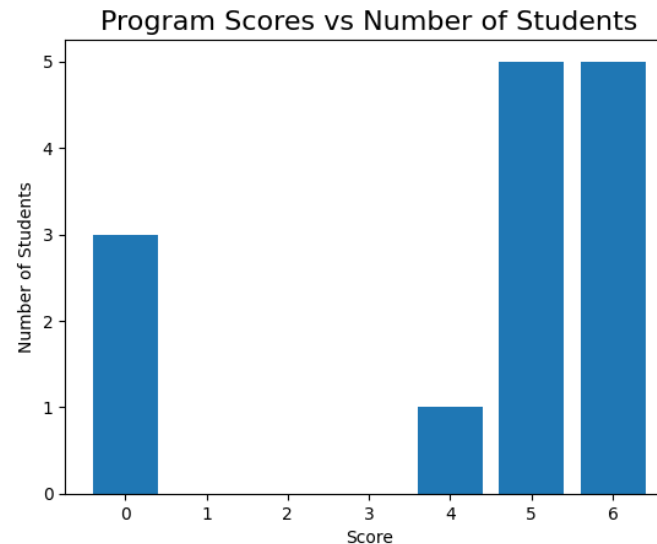
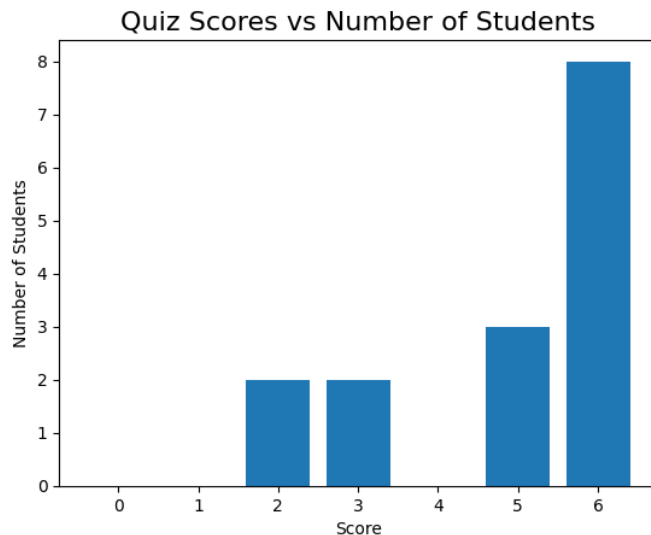
- Evaluate for 'n' = 2 , 4, 5, 6
- Follow up reading (optional): [nested functions visualized](#)

Programming, , Test #1

- Need to know
 - Data types
 - Conditionals : If/else
 - Loops : for
 - Arrays : 2 dimensional
- Practice
 - [Arrays- DS \(Hackerrank\)](#)
 - Do the 4 problems marked “easy”
 - Optional : “hard”, “medium” one are for extra challenge

Test 1, observations

- Great participation, nearly everyone attempted!
- Overall students did better in quiz
 - 8x full marks in quiz vs 5x full marks in programming
- 4 students got a zero in programming (with 1 not attempted)



Test 2 , analysis

- The quizzes
 - seem to be easier for students to master
 - Success cause everyone attempted, next we figure out how to raise scores there
 - Suggestions ?
- Programming
 - It was amazing attempt
 - The codes generally worked
 - But 4/15 students had zeroes.
 - Suggestions ?
- Goal :We need raise scores for everyone and move some in attempted stage
 - Note
 - Compete with yourself , Do better than your last attempt
 - Ask for help
 - on what you don't know, to learn it.
 - And even on what you know , i.e. to master it

Topics for Test 2

- Mathematical expressions
 - Representations (infix, prefix, postfix)
 - Evaluation or expression
- Discrete Mathematics
 - Logical Operators (AND , OR , NOT, SHIFT,...)
 - Boolean logic
- Programming
 - char, string ,arrays
 - Loops
 - Conditionals
 - If/else
 - switch/case

Infix expressions

- Encounter them in grade maths. e.g.
 - $(11+14)/(9-3) + 2$
 - $3+7 / (4 * 5 - 6)$
 - $- 2 + 8 / 2$
- Can also be written using variables
 - $(a + b) / (c - d) + e$
 - $x + y / (g * h - k)$
- Evaluated using
 - PEMDAS / BODMAS

Expression : anatomy

- Expression is made of
 - Operators (+ , - , * , /)
 - Operands (numbers, variables)
- Unary operators : - a
 - <operator> <operand>
- Binary operators (infix) : a + b
 - <operand> <operator> <operand>

Prefix/Postfix expressions

- Benefits
 - Remove ambiguity(i.e. can forget PEMDAS, BODMAS)
 - Computer friendly
 - Works with both unary/binary
 - Faster evaluation
- Expression styles
 - Infix : operator in **middle** of operands
 - $(5+6)*3$
 - $\langle \text{operand} \rangle \langle \text{operator} \rangle \langle \text{operand} \rangle$
 - Prefix : operator **before** operands
 - $* + 5 6 3$
 - $\langle \text{operator} \rangle \langle \text{operand} \rangle \langle \text{operand} \rangle$
 - Postfix : operator **after** operands
 - $5 6 + 3 *$
 - $\langle \text{operand} \rangle \langle \text{operand} \rangle \langle \text{operator} \rangle$

Evaluating expression, *but first **Stacks and Queues***

- What is a **Stack** ?
 - This is a sequential structure of **items**
 - That are **pushed at one end**
 - **Pulled** via the **same end**
 - **LIFO** : Last in First Out
 - Examples
 - Stack of plates
 - Your turned in paper assignments
- What is a **Queue**?
 - Another sequential structure of **items**
 - that are **pushed at one end**
 - **Pulled** via the **opposite end**
 - **FIFO** : First In First Out
 - Examples
 - Queue for buying tickets
 - Traffic in one way single lane

Evaluating **prefix** expression

- Infix : $(5+6)*3$
 - $* + 5 6 3$
 - $- + 2 * 3 4 / 16 ^ 2 3$
- Algorithm (harder way)
 - Scan from right to left
 - Anytime you find an operator
 - Evaluate the operation using the previous 2 operands
 - Replace the **<operator>** **<operand>** **<operand>** with **result** in the expression
 - Repeat the steps till only 1 operand is left

Evaluating postfix expression

- Evaluate
 - $2\ 3\ 1\ * + 9\ -$
 - $5\ 3 + 6\ 2 / * 3\ 5 * +$
- *Algorithm to evaluate (use **stack**)*
 1. Push the operands in a stack
 2. When you encounter a operator
 - Pop 2 operands
 - Perform the operation on operands
 - Push the result in stack
 - Repeat from 1 until expression is parsed
 - The last item in stack is the answer
 - There would be only 1 item left when done correctly
- * prefix expr can also be evaluated in this way (*just in **reverse***)

Evaluate

- Prefix

- $- * 5 + - 4 2 2 / 6 3$
- $- * + 3 5 7 + / 4 2 1$
- $- + 10 * 2 3 + 4 / 5 5$

- Postfix

- $1 2 + 3 4 + * 5 6 - / 7 +$
- $8 2 / 3 4 + * 5 1 + 2 / -$
- $9 8 4 2 1 ^ * / - 3 +$

Answers

- Prefix

- $- * 5 + - 4 2 2 / 6 3$
 - Ans : **18**
- $- * + 3 5 7 + / 4 2 1$
 - Ans : **53**
- $- + 10 * 2 3 + 4 / 5 5$
 - Ans : **11**

- Postfix

- $1 2 + 3 4 + * 5 6 - / 7 +$
 - Ans : **-14**
- $8 2 / 3 4 + * 5 1 + 2 / -$
 - Ans : **25**
- $9 8 4 2 1 ^ * / - 3 +$
 - Ans : **11**

Practice: Programming

- Write a function that takes a string and checks if it is a palindrome
 - Returns true if palindrome is found else false
 - Hints: string indexing, loops, if , comparing characters
- Count the number of vowels, consonants in a sentence.
 - Vowels : a,e,i,o,u
 - Consonant : everything else other than vowels
 - Ignore : spaces (' ',) comma(,), dash(-),semicolon(;),colon(:)
- Find the most frequent word in a sentence
 - If more than 1 word has same frequency return the lexicographically smaller one.

Review Evaluating Prefix/Postfix

- From last week
 - Operator
 - Operand
 - Stack
 - Queue
- Evaluating Prefix/Postfix
 - <3 questions each>

Precedence and associativity

- Precedence (highest priority first)
 - $()$
 - $^$
 - $/$ $*$
 - $-$ $+$
- Associativity
 - Right to left : $^$
 - Left to right : $+$, $-$, $*$, $/$
- Notes
 - $()$ not categorized as associative, they represent grouping mechanism

Precedence and associativity

- Example

- $10 + 20 / 2 * 3 - 5$

- $\rightarrow (10 + 20) / 2 * (3 - 5)$ // added parenthesis for denoting precedence

- $\rightarrow 30 / 2 * -2$

- $\rightarrow (30 / 2) * -2$ // left to right precedence in action for $/, *$ (as they are equal)

- $\rightarrow -30$

Precedence and associativity (lets evaluate)

- $2 + 3 * 4 ^ 2$
- $2 ^ 3 ^ 2$
- $100 / 2 * 5 ^ 2 ^ 1$

Precedence and associativity (answers)

- $2 + 3 * 4 ^ 2$
 - $\rightarrow 2 + 3 * 8$
 - $\rightarrow 2 + 24$
 - $\rightarrow 26$
- $2 ^ 3 ^ 2$
 - $\rightarrow 2 ^ 9$
 - $\rightarrow 512$
 - ~~Not 64~~
- $100 / 2 * 5 ^ 2 ^ 1$
 - $\rightarrow 100 / 2 * 25$
 - $\rightarrow 50 \times 25$
 - $\rightarrow 1250$

Convert to Prefix/Postfix (examples)

- $B * C$
 - Prefix : *** B C**
 - Postfix : **B C ***
- $A + B * C$
 - Prefix : **+A*BC**
 - Postfix : **ABC*+**
- $(5 + 6) * 3$
 - Prefix : *** + 5 6 3**
 - Postfix : **5 6 + 3 ***

Convert to Postfix

- How to convert!
 - Convert the highest priority first
 - (and parantheseize for mental ease)
- $(a + b) / (c - d) + e$
- $x + y / (g * h - k)$

Convert to Postfix

- How to convert!
 - Convert the highest priority first
 - (and parantheseize for mental ease)
- $(a + b) / (c - d) + e$
 - Prefix : **$+ / a b - c d e$**
 - Postfix : **$a b + c d - / e +$**
- $x + y / (g * h - k)$
 - Prefix : **$+ x / y - * g h k$**
 - Postfix : **$x y g h * k - / +$**

Convert to Prefix and Postfix

- $a + b - c * d + e ^ f$
- $(a + b) * c - (d - e) * (f + g)$
- $((a + b) * (c + d) / (e - f)) + g$
- $a * (b + c) / (d - e)$
- $a - b / (c * d ^ e)$
- $(a + b) * (c + d) - e$

Convert to Prefix and Postfix (answers)

Infix	Prefix	Postfix
$a + b - c * d + e ^ f$	$+ - + a b * c d ^ e f$	$a b + c d * - e f ^ +$
$(a + b) * c - (d - e) * (f + g)$	$- * + a b c * - d e + f g$	$a b + c * d e - f g + * -$
$((a + b) * (c + d) / (e - f)) + g$	$+ / * + a b + c d - e f g$	$a b + c d + * e f - / g +$
$a * (b + c) / (d - e)$	$/ * a + b c - d e$	$a b c + * d e - /$
$a - b / (c * d ^ e)$	$- a / b * c ^ d e$	$a b c d e ^ * / -$
$(a + b) * (c + d) - e$	$- * + a b + c d e$	$a b + c d + * e -$

Logical operators (for bits/bytes)

- Bitwise
 - NOT, AND, XOR , OR
- Shift
 - Linear : LSHIFT, RSHIFT,
 - Circular : RCIRC, LCIRC
- Precedence
 - NOT,
 - SHIFT, CIRC,
 - AND ,
 - XOR,
 - OR

Truth Tables

X	NOT X
0	1
1	0

X	Y	X AND Y	X OR Y	X XOR Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Truth Tables

- Linear shift , add 0s as padding (either left or right)
 - Enough of linear shifts would result in all 0s
- Circular shift just rolls over the bits (as per direction)
 - Shifting by the number of word size , same bit sequence would be obtained

X	(LSHIFT-2 X)	(RSHIFT-3 X)	(LCIRC-3 X)	(RCIRC-1 X)
01101	10100	00001	01011	10110
10	00	00	01	01
1110	1000	0001	0111	0111
1011011	1101100	0001011	1011101	1101101

Operations (evaluate)

- 0110 AND 1011
- 1100 OR 0101
- 1010 XOR 1111
- 1010 AND 1100 OR 0001
- 0011 OR 1010 AND 0110
- (1100 XOR 1010) AND 0110