**CSF FINAL EXAM SOLUTIONS OF 00009115**

**Task 1**

<https://github.com/00009115/CSF.CW1.00009115/tree/exam>

**Task 2**

1. Dividing 9115 by 2 until the quotient is 0 to convert into *binary*:

|  |  |
| --- | --- |
| **Quotient**  9115 | **Remainder**  1 |
| 4557 | 1 |
| 2278 | 0 |
| 1139 | 1 |
| 569 | 1 |
| 284 | 0 |
| 142 | 0 |
| 71 | 1 |
| 35 | 1 |
| 17 | 1 |
| 8 | 0 |
| 4 | 0 |
| 2 | 0 |
| 1 | 1 |
| 0 |  |

911510 = 100011100110112

Dividing 9115 by 16 until the quotient is 0 to convert into *hex*:

|  |  |
| --- | --- |
| **Quotient**  9115 | **Remainder**  11 |
| 569 | 9 |
| 35 | 3 |
| 2 | 2 |
| 0 |  |

1110 = B16

911510 = 239B16

1. Dividing 9999 by 2 until the quotient is 0 to convert into *binary*:

|  |  |
| --- | --- |
| **Quotient**  99999 | **Remainder**  1 |
| 49999 | 1 |
| 24999 | 1 |
| 12499 | 1 |
| 6249 | 1 |
| 3124 | 0 |
| 1562 | 0 |
| 781 | 1 |
| 390 | 0 |
| 195 | 1 |
| 97 | 1 |
| 48 | 0 |
| 24 | 0 |
| 12 | 0 |
| 6 | 0 |
| 3 | 1 |
| 1 | 1 |
| 0 |  |

9999910 = 110000110100111112

911510 = 100011100110112

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
|  |  |  | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
|  |  | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

11000011010011111 + 10001110011011 = 11010101000111010

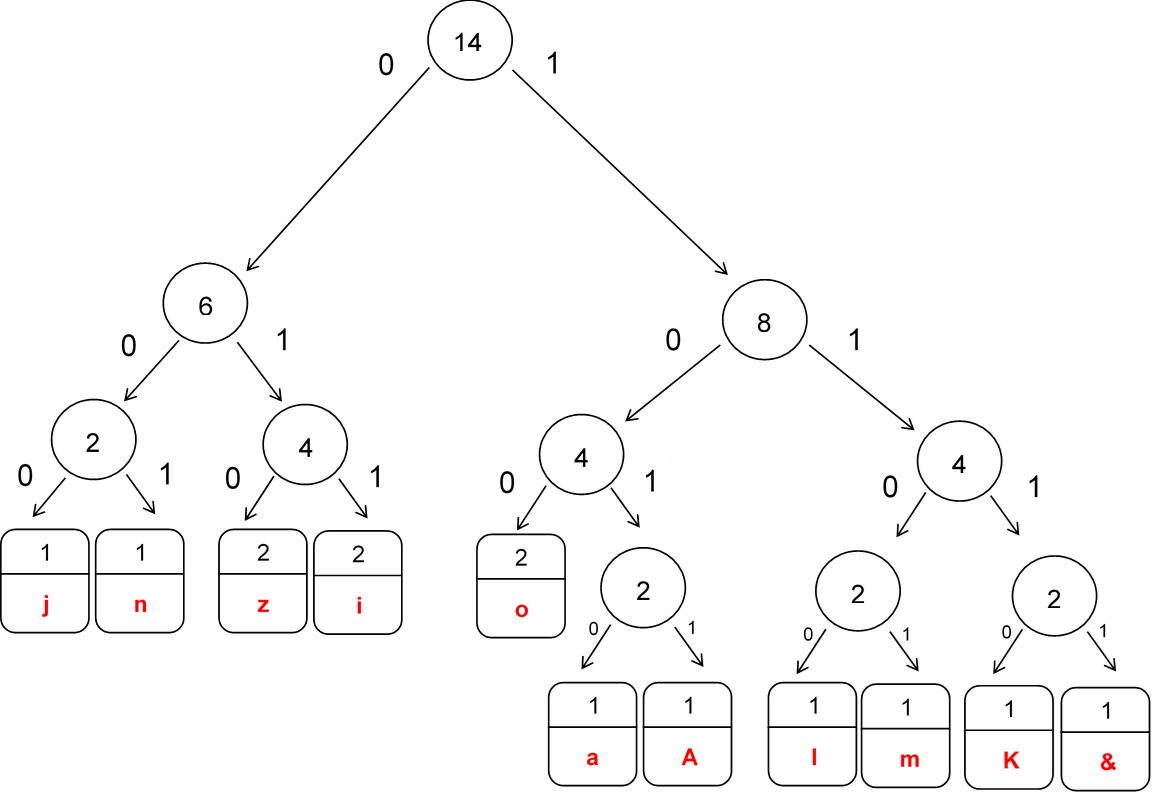
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
|  |  |  | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
|  |  | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

11000011010011111 - 10001110011011 = 10110001100000100

1. Hexadecimal or Hex numbers are widely used to denote colors in the so-called “hex” scheme. This is preferred over binary numbers because it is cleaner and can be defined with fewer digits. And the reason for choosing hex numbers over decimal numbers is that hex numbers have a base which is a multiple of 2 which can be used as a connection between hex and binary which decimal numbers do not have. This is important because hex numbers will be converted into the machine language – binary numbers at the end of the day.

**Task 3**

Aziza&Komiljon



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Character** | **Frequency** | **Code** | **Single Size (Bits)** | **Total size (Bits)** |
| z | 2 | 010 | 3 | 6 |
| i | 2 | 011 | 3 | 6 |
| o | 2 | 100 | 3 | 6 |
| j | 1 | 000 | 3 | 3 |
| n | 1 | 001 | 3 | 3 |
| a | 1 | 1010 | 4 | 4 |
| A | 1 | 1011 | 4 | 4 |
| l | 1 | 1100 | 4 | 4 |
| m | 1 | 1101 | 4 | 4 |
| K | 1 | 1110 | 4 | 4 |
| & | 1 | 1111 | 4 | 4 |
| TOTAL | 14 | - | 39 | 48 |

Encoded message: 101101001101010101111111010011010111100000100001

48 Bits in total

**Task 4**

Unordered: 9, 1, 1, 5, 4, 5, 2, 3, 7

Ordered: 1, 1, 2, 3, 4, 5, 5, 7, 9

The number to look for is: x=**2**

1) Midpoint is 4 (9/2=5.5, => 5th digit is the midpoint)

2) x<midpoint (2<4, ignore the range on the right: 1, 1, 2, 3, ~~4, 5, 5, 7, 9~~)

3) Midpoint is 1 (4/2=2 => 2nd digit in the remaining set is the midpoint)

4) x>midpoint (2>1, ignore the range on the left: ~~1, 1,~~ 2, 3)

5) Midpoint is 2 (2/2=1 => 1st digit in the remaining set is the midpoint)

6) x=midpoint (2==2) The number is found.

**Task 5**

Paged memory management uses fixed-size blocks which are called “pages” unlike other methods dividing the memory into various parts. In this methos, main memory is divided into same-sized blocks or frames where the data will be placed eventually. This method is used in order to achieve fast accessing processes and increasing the efficiency of the logical concept.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Page | 0 | 1 | 2 | 3 | 4 |
| Frame | 4 | 3 | 5 | 6 | 7 |

<2, 85> on the page of 2 we have **5** frames

We should calculate 5 frames of 1024-sized each + offset: 5 \* 1024 + 85 = 5205

1. <0, 1026> this logical address is invalid since it exceeds the frame size of 1024 (1024<1026)