1. <https://github.com/00010401/CSF.CW1.00010401>
2. a) ID number: 10401.

Converting ID number in decimal into its binary equivalent:

|  |  |  |
| --- | --- | --- |
|  | **Remainder** | **Binary** |
| 10401/2=5200 | 1 | 1 |
| 5200/2=2600 | 0 | 0 |
| 2600/2=1300 | 0 | 0 |
| 1300/2=650 | 0 | 0 |
| 650/2=325 | 0 | 0 |
| 325/2=162 | 1 | 1 |
| 162/2=81 | 0 | 0 |
| 81/2=40 | 1 | 1 |
| 40/2=20 | 0 | 0 |
| 20/2=10 | 0 | 0 |
| 10/2=5 | 0 | 0 |
| 5/2=2 | 1 | 1 |
| 2/2=1 | 0 | 0 |
| 1/2=0 | 1 | 1 |

So, the number is 10100010100001.

Converting ID number in decimal to its hex equivalent:

|  |  |  |
| --- | --- | --- |
|  | **Remainder** | **Hex** |
| 10401/16=650 | 1 | 1 |
| 650/16=40 | 10 | A |
| 40/16=2 | 8 | 8 |
| 2/16=0 | 2 | 2 |

So, the number is 28A1.

b) 1)

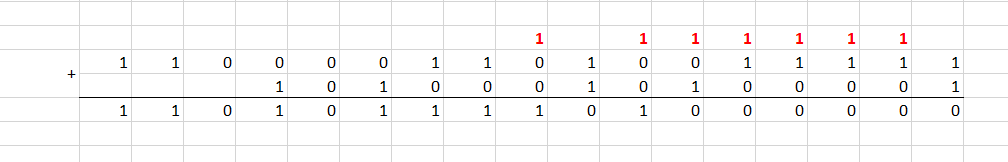
Decimal 99999 to binary:

|  |  |  |
| --- | --- | --- |
|  | **Remainder** | **Binary** |
| 99999/2=49999 | 1 | 1 |
| 49999/2=24999 | 1 | 1 |
| 24999/2=12499 | 1 | 1 |
| 12499/2=6249 | 1 | 1 |
| 6249/2=3124 | 1 | 1 |
| 3214/2=1562 | 0 | 0 |
| 1562/2=781 | 0 | 0 |
| 781/2=390 | 1 | 1 |
| 390/2=195 | 0 | 0 |
| 195/2=97 | 1 | 1 |
| 97/2=48 | 1 | 1 |
| 48/2=24 | 0 | 0 |
| 24/2=12 | 0 | 0 |
| 12/2=6 | 0 | 0 |
| 6/2=3 | 0 | 0 |
| 3/2=1 | 1 | 1 |
| 1/2=0 | 1 | 1 |

So, 99999 in decimal is equal to 11000011010011111 in binary.

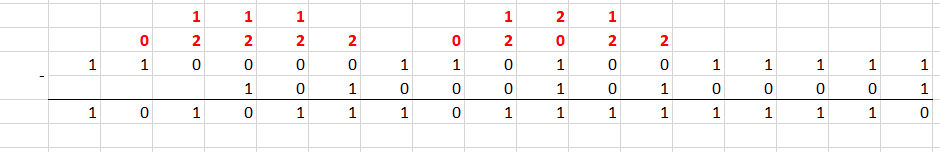
ID 10401 in decimal is equal to 10100010100001 in binary.

Decimal 99999 in binary and ID number in binary addition:



**Carry ->**

2) Decimal 99999 in binary and ID 10401 in binary subtraction:



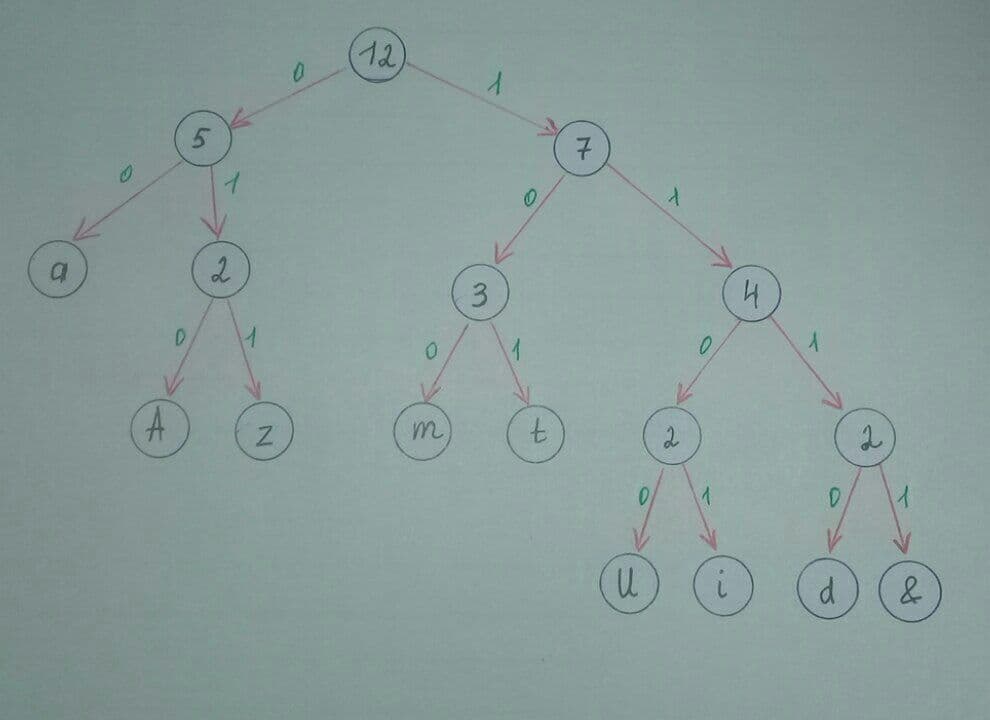
**<- Borrow**

c) Hex number system is mainly used by programmers, as it eases binary system by allowing grouping of binary digits and demonstrating long binary values in a short format, which makes it easy to understand compared to long binary values. It handles more information, however uses comparatively less storage. Moreover, hex numbers are used to represent colors in HTML and CSS; MAC addresses of devices; memory location of the errors. In hex number system it is easier to track the bugs and to debug, also it decreases the bug occurrence. (1)

1. Umida&Azamat
2. Table:

|  |  |  |
| --- | --- | --- |
| Letter | Frequency | Encoding |
| U | 1 | **1100** |
| m | 2 | **100** |
| i | 1 | **1101** |
| d | 1 | **1110** |
| a | 3 | **00** |
| & | 1 | **1111** |
| A | 1 | **010** |
| z | 1 | **011** |
| t | 1 | **101** |

1. Huffman tree:



Answer: 37bits will be needed to encode “Umida&Azamat” message using Huffman encoding.

1. a) 10401 and 45237

1, 0, 4, 0, 1, 4, 5, 2, 3, 7

Sorting: 0, 0, 1, 1, 2, 3, 4, 4, 5, 7

b) The number to look for: 5

c) Pseudo code:

1) Find midpoint:

a) Count the total number of entries in the set of numbers; (10)

b) Divide it by 2; (10/2=5)

c) 5th number will be midpoint. (2)

2) If ignore the range on the left till the midpoint, inclusively. Otherwise, ignore the range on the right side, beginning with midpoint. (As 5>2, we ignore [0-2]).

3) Find midpoint from remaining set of numbers:

a) Count the total number of entries in the set of numbers; (5)

b) Divide it by 2; (5/2=2.5)

c) Round the result 2.5≈3.

d) 3rd number will be midpoint. (4)

4) If ignore the range on the left till the midpoint, inclusively. Otherwise, ignore the range on the right side, beginning with midpoint. (As 4<5, we ignore [3-4]).

5) Find midpoint (2/2=1). 1st value is 5.

6) (x=5) = (midpoint=5), so the number is found.

1. Main memory in paged memory management consists of small fixed-size frames, and process includes pages which have an equal size with frames. In contrast, partition management technique is made up from various blocks/partitions which are mostly stored in contiguous areas of memory. (2)

In single contiguous memory management, the program should find one large piece of memory for loading data, however, in paged management technique it will search for smaller chunks of memory to load data, as it allows to divide the process into small parts.

Moreover, in all other memory management techniques the whole process of loading program should place in memory entirely, but the demand paged technique gives an opportunity to bring only needed parts of process into memory and to bring others parts when needed by swapping between main and secondary memory pages. (2)



To find physical address, according to formula we need number of frames. We can find number of frames by looking to the given table in cell respective to the page number. So, as in this case page number is 2, the number of frames will be 5 respectively.



Answer: We cannot find physical address in this case, as given offset of page overflows the frame size.

Reference list:

1. Uses of Hexadecimal [Internet]. Computer Science. 2019 [cited 2021 Jan 14]. Available from: <https://www.computerscience.gcse.guru/theory/uses-of-hexadecimal>
2. Dale N, Lewis J. Computer science illuminated. 7th ed. Burlington, Massachusetts: Jones & Bartlett Learning; 2020.

‌

‌