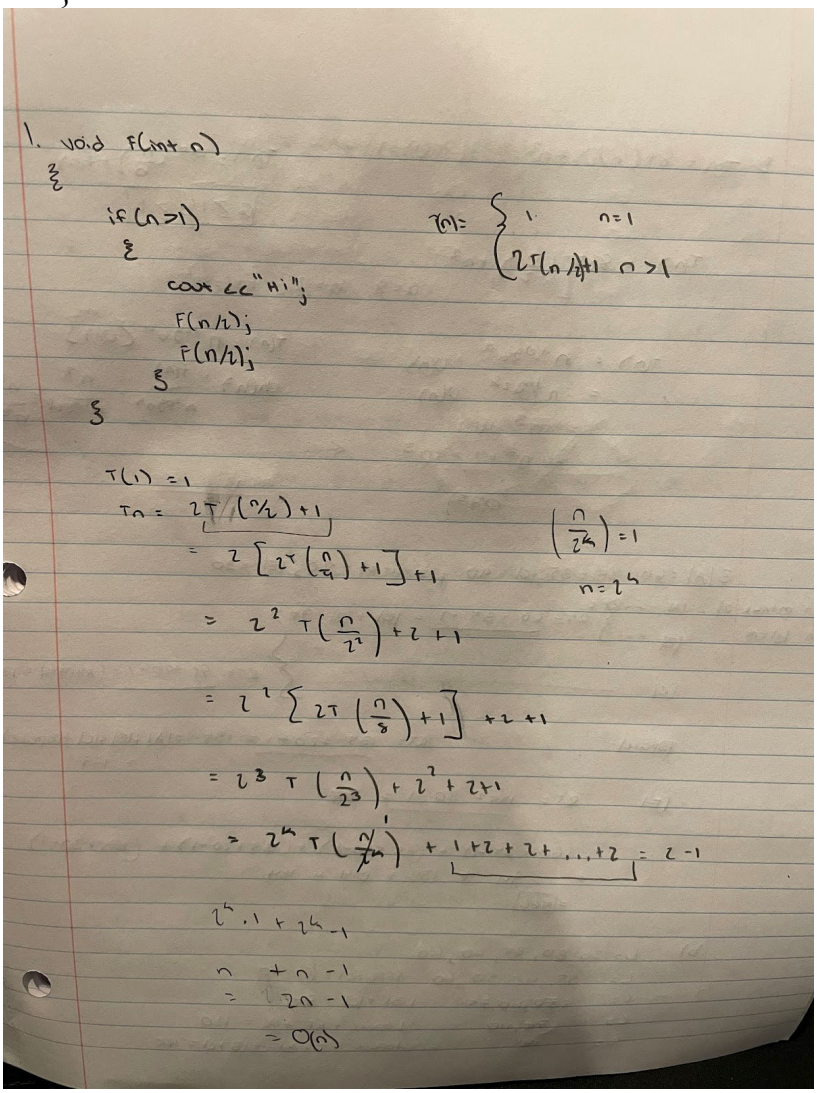
CPSC 335
Assignment No. 2 Name Nicholas Ayson
80 points

1. Find the run-time of this recursive function by computing the complexity of the recurrence equation below.

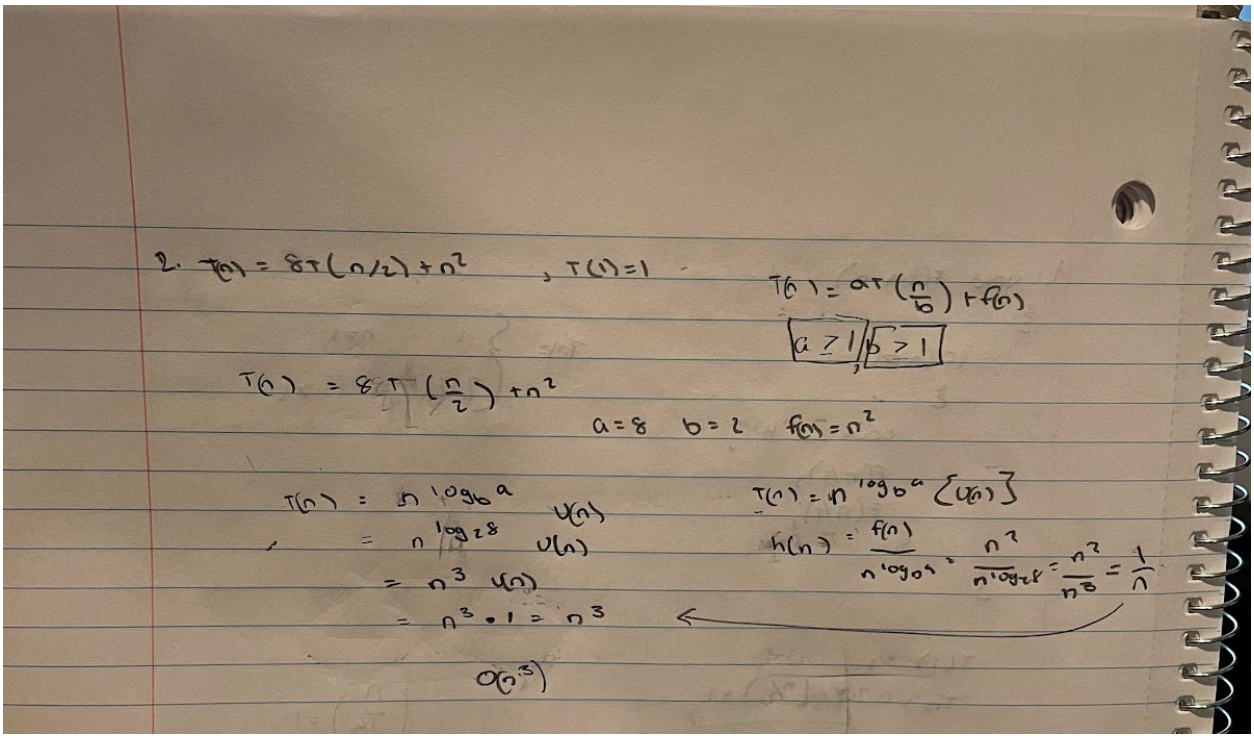
```
void F(int n)
{
    if( n>1)
    {
       cout<<"Hi";
       F(n/2);
       F(n/2);
    }
}
```

$$T(n)=\begin{cases} 1 & n=1 \\ 2T(n/2)+1 & n>1 \end{cases}$$



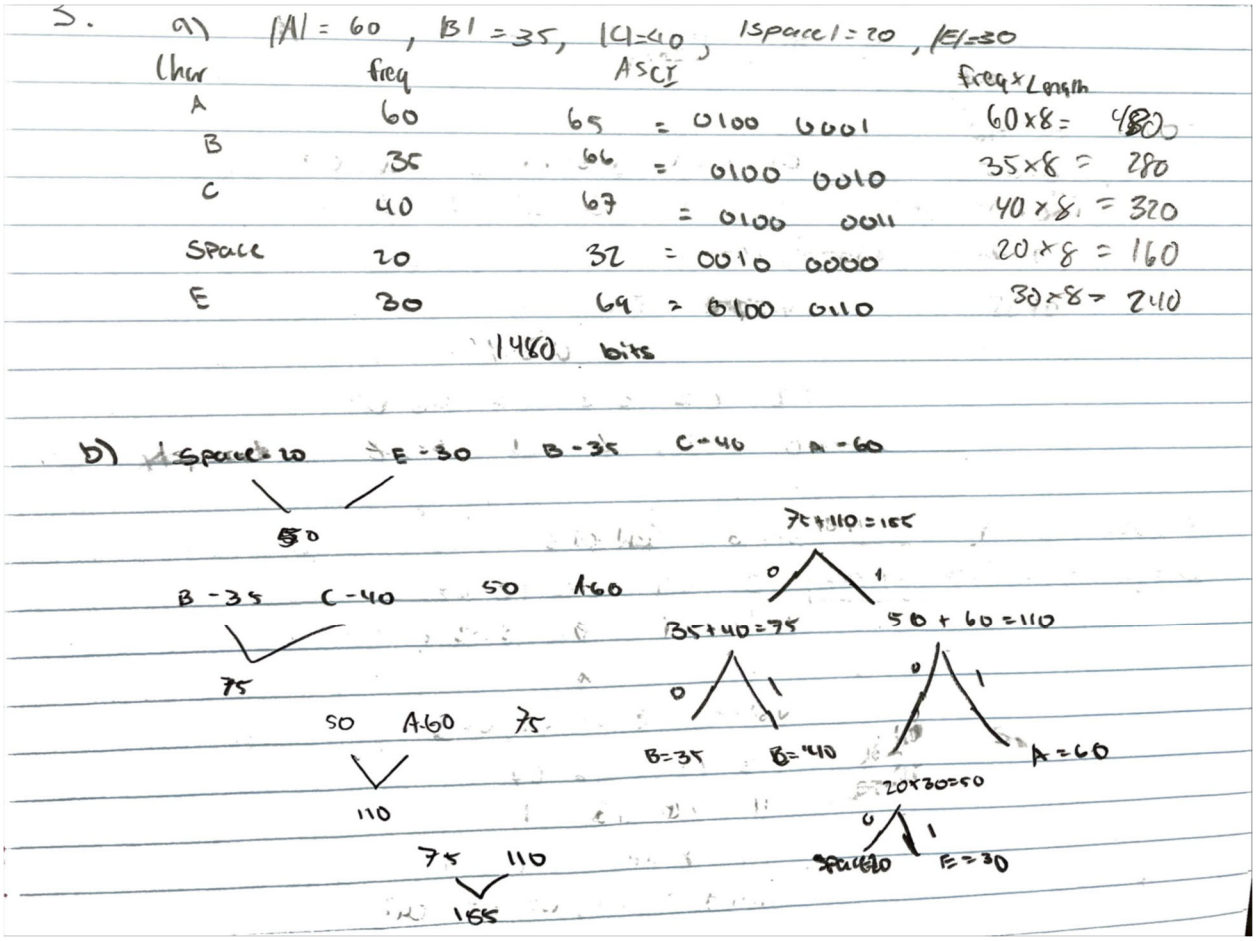
2. Find the complexity of the following recurrence relation

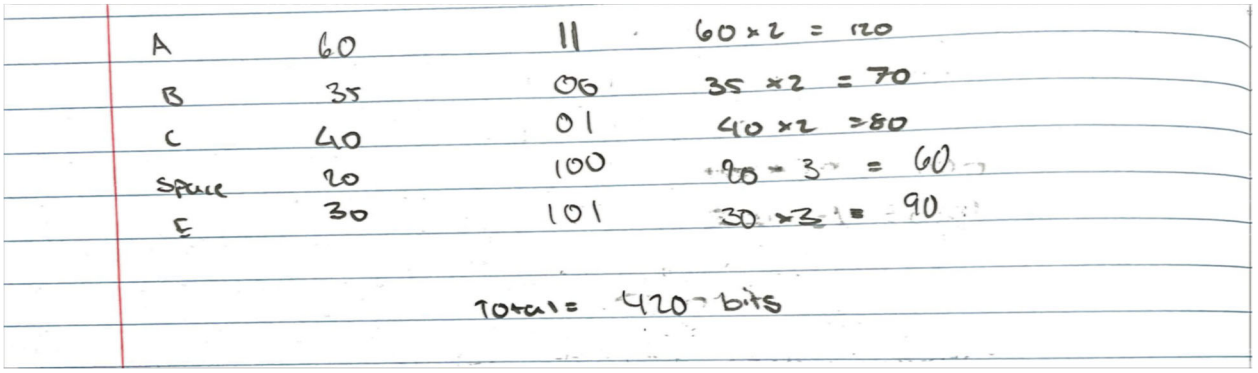
$T(n) = 8T(n/2) + n^2,\ T(1) = 1$



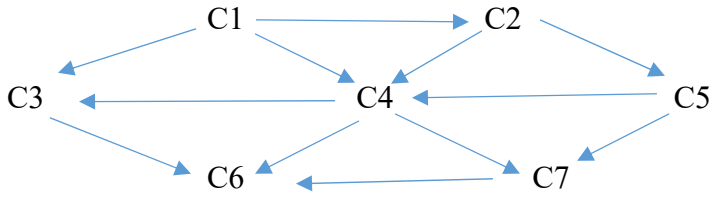
3. Given a file in which |A|=60, |B|=35, |C|=40, |space|=20, and |E|=30.

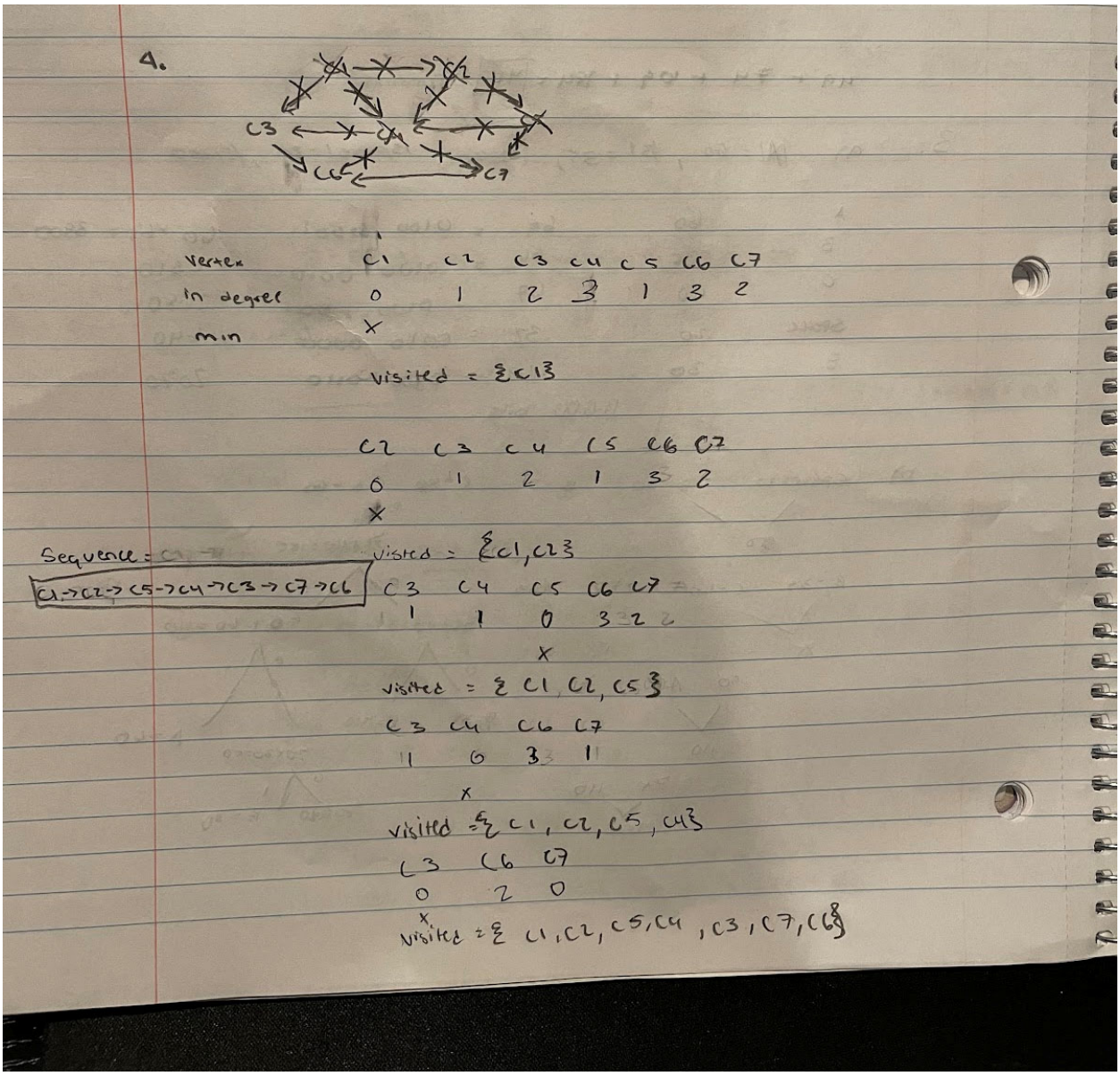
   a. How many bits are used to store the file?
   b. Use Huffman Coding compression algorithm to find a new binary code for each letter. Use the new binary code to find the number of bits to store the file.





4. For the following directed graph which represents list of courses and their prerequisites. Determine in what order a student can take all courses and meet all prerequisite's requirements. Us the topological sorting algorithm.





5. Solve the following knapsack problem using ***set methods***

Items	X1	X2	X3	X4
Value	40	30	50	10
Weight	2	5	10	5

Maximum weight that can be hold by the sack is 16 pounds

6. Let S ={ a,b,c,d,e,f,g} be a collection of different objects with a given value and weight. Use ***fractional knapsack algorithm*** to fill a sack with maximum capacity of weight=18

Objects	a	b	c	d	e	f	g
Value	12	10	8	11	14	7	9
Weight	4	6	5	7	3	1	6

5. Items x1 x2 x3 x4 Max = 16

value 40 30 50 10

weight 2 5 4 5

$S_0 = \{(0,0)\}$

$S_1 = S_0 + (40,2) = \{(40,0)\}$

$S_2 = \{(40,2), (30,5), (40+30, 2+5)\} = \{(40,2), (30,5), (70,7)\}$

$S_3 = \{(40,2), (30,5), (70,7), (50,10), (90,12), (100,12), (80,15), (110,9)\}$

$S_4 = \{(40,2), (30,5), (70,7), (50,10), (90,12), (100,12), (80,15), (120,17), (110,15)\}$

(50,7) (150,22) (130,22) (150,22) ...

x3 and x2 (80,15) − (50,10) = (30,5)

x3 and x2 are items in the sack

6. S = { a b c d e f g } fraction knapsack algo

	x	x	x	x	x	x	x
objects	a	b	c	d	e	f	g
value	12	10	8	11	14	7	9
weight	4	6	5	7	3	1	6
	12/4 = 3	1.7	1.6	1.6	4.7	7	1.5
selected	1	1	1	0	1	0	0

max = 18

steps 1. Sack { e } c = 18 − 3 = 15, 4.7

Sack = { e, g } c = 15 − 9 = 11, 4.7 + 3 = 7.7

Sack = { e, g, b } c = 11 − 6 = 5, 7.7 + 1.7 = 9.4

Sack = { e, g, b, c } c = 5 − 5 = 0, 9.4 + 1.6 = 11

Value = 11

Programming assignments

1. Given array : int a[10]; Write a program to generate 10 random numbers between 1 and 6 and
   - store them in array a. Display array a, compute and display Mean, Median, and the Mode of data in array a.
     It is possible the set of data have more than one mode (two or more numbers have the same high frequency), display all modes

```
// 1.   Given array : int a[10];  Write a program to generate 15 random numbers
between 1 and 6 and store them in array a.
// Display array a, compute and display Mean, Median, and the Mode of data in
array a.
// It is possible the set of data have more than one mode ( two or more numbers
have the same high frequency), display all modes

#include <bits/stdc++.h>
using namespace std;

// Function for calculating mean
double findMean(int a[], int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += a[i];

    return (double)sum / (double)n;
}

// Function for calculating median
double findMedian(int a[], int n)
{
    // check for even case
    if (n % 2 != 0){
        return (double)a[n / 2];
    }

    return (double)(a[(n - 1) / 2] + a[n / 2]) / 2.0;
}

//function to calculate mode and print itself
void findMode(int a[], int n)
{
    int y[10]={};
    int i, j, k, m, cnt, count, max=0;
    int mode_cnt=0;
    int num;
    int v;
```

```
    vector<int> results;
    vector<int>::iterator pos;

    //loop to count an array from left to right
    for(k=0; k < 10; k++) {
      cnt=0;
      num=a[k]; //num will equal the value of x[k]
      for(i=k; i<10; i++) {
         if(num==a[i])
            cnt++;
      }
      y[k]=cnt;
    }

    //find highest number in array
    for(j=0; j<10; j++)
    {
      if(y[j]>max)
         max=y[j];
    }

    //find how many modes there are
    for(m=0; m<10; m++)
    {
      if(max==y[m])
         mode_cnt++;
    }

    //push results to vector
    for (m=0; m < 10; m++)
    {
      if(max == y[m])
      {
        //after taking out this line the code works properly
        //      std::cin >> x[m];
        results.push_back(a[m]);
      }
    }
    //sort vector and print
   std::sort(results.begin(), results.end());
   cout << "The mode(s) is/are: ";
   for (pos=results.begin(); pos!=results.end(); ++pos) {
         cout << *pos << " ";
   }
}
```

```
// Driver code
int main()
{
    //given
    int a[10];
    int n = sizeof(a) / sizeof(a[0]);

    srand(time(0));//help with rand function

    for(int i=0; i < n; i++){
        a[i]= rand() % 6 + 1;  //Generate number between 1 to 6
    }

    cout<<"\nElements of the array:"<<endl;
    for(int i=0; i < n; i++){
        cout<<"Elements no "<<i+1<<":"<<a[i]<<endl;
    }

    // Function call
    cout << "Mean = " << findMean(a, n) << endl;
    cout << "Median = " << findMedian(a, n) << endl;
    findMode(a , n);
    return 0;
}
```

OUTPUT:

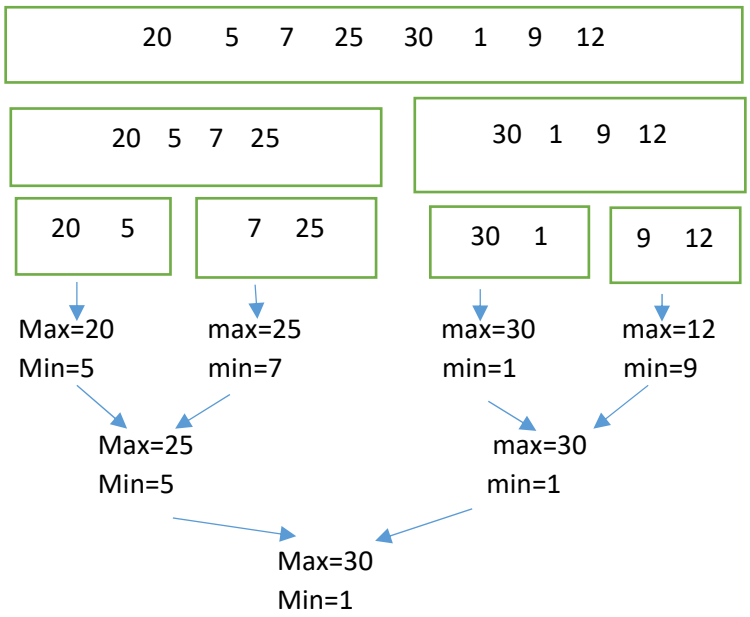
```
Running] cd "c:\Users\Nick\Desktop\Algorithm-Engineering-Work\randomnumbers\" &&
g++ random.cpp -o random && "c:\Users\Nick\Desktop\Algorithm-Engineering-
Work\randomnumbers\"random

Elements of the array:
Elements no 1:2
Elements no 2:6
Elements no 3:2
Elements no 4:2
Elements no 5:4
Elements no 6:3
Elements no 7:4
Elements no 8:4
Elements no 9:3
Elements no 10:3
Mean = 3.3
Median = 3.5
The mode(s) is/are: 2 3 4
[Done] exited with code=0 in 1.093 seconds
```

```
[Running] cd "c:\Users\Nick\Desktop\Algorithm-Engineering-Work\randomnumbers\" &&
g++ random.cpp -o random && "c:\Users\Nick\Desktop\Algorithm-Engineering-
Work\randomnumbers\"random

Elements of the array:
Elements no 1:1
Elements no 2:1
Elements no 3:5
Elements no 4:4
Elements no 5:6
Elements no 6:4
Elements no 7:2
Elements no 8:2
Elements no 9:5
Elements no 10:3
Mean = 3.3
Median = 5
The mode(s) is/are: 1 2 4 5
[Done] exited with code=0 in 1.106 seconds
```

2. Use **<u>divide-and-conquer algorithm</u>** to find the maximum and the minimum data in array:
   int a[8]={20,5,7,25,30,1,9,12};




```
#include <iostream>
#include <algorithm>

using namespace std;

struct node {
```

```
    int min;
    int max;

    void print() const { cout << "Max: " << max << "\n" << "Min: " << min <<
"\n\n"; }
};

node minmax(int a[], int s, int n)
{
    if (n-s > 2)
    { // array size contains at least 2 items
        node b = minmax(a, s, (n + s)/2); b.print();
        node c = minmax(a, (n + s)/2, n); c.print();
        return { min(b.min, c.min), max(b.max, c.max) };
    }
    return { min(a[s], a[n-1]), max(a[s], a[n-1]) };
}

int main()
{
    int a[8] = { 20, 5, 7, 25, 30, 1, 9, 12 }; //given
    node result = minmax(a, 0, 8);
    result.print();
    return 0;
}
```

OUTPUT:

```
[Running] cd "c:\Users\Nick\Desktop\Algorithm-Engineering-Work\minmax\" && g++
minmax.cpp -o minmax && "c:\Users\Nick\Desktop\Algorithm-Engineering-
Work\minmax\"minmax
Max: 20
Min: 5

Max: 25
Min: 7

Max: 25
Min: 5

Max: 30
Min: 1

Max: 12
Min: 9

Max: 30
```

```
Min: 1

Max: 30
Min: 1


[Done] exited with code=0 in 0.428 seconds
```