KIIT Deemed to be University
Online Mid Semester Examination(Autumn Semester-2021)

DAA SOLUTION & EVALUATION SCHEME-Final

Subject Name & Code: Design & Analysis of Algorithms (CS-2012)
Applicable to Courses: CSE, IT, CSCE,CSSE & ECS

Full Marks=20 **Time:1 Hour**

SECTION-A(Answer All Questions. All questions carry 2 Marks)

Time:20 Minutes **(5×2=10 Marks)**

Question No	Question Type (MCQ/ SAT)	Question	Answer Key (if MCQ)	CO Mapping

Q.No: 1(a)	MCQ	Consider the following code fragment.
		int a[] = {2, 1, 3, 4, 5, 6, 7, 8, 9, 0}
		int fun(int b[], int n)
		{
		if (n==1)
		return b[n-1];
		else return b[n] + fun(b, n-1) + fun(b, n-1);
		}
		What is the result of the function call fun(a, 3) and What's the asymptotic running time of this function in terms of n respectively.
		A. 18, $\Theta(n)$
		B. 18, $\Theta(2^n)$
		C. 19, $\Theta(n)$
		D. 19, $\Theta(2^n)$
		E. NONE
			B	CO1

	MCQ	Consider the following code fragment.
		int a[] = {2, 1, 3, 4, 5, 6, 7, 8, 9, 0}
		int fun(int b[], int n)
		{
		if (n==1)
		return b[n-1];
		else return b[n] + 2*fun(b, n-1);
		}
		What is the result of the function call fun(a, 3) and What's the asymptotic running time of this function in terms of n respectively.
		A. 18, $\Theta(n)$
		B. 18, $\Theta(2^n)$
		C. 19, $\Theta(n)$
		D. 19, $\Theta(2^n)$
		E. NONE
			A	CO1

	MCQ	Consider the following code fragment.	D	CO1

		int a[] = {2, 1, 4, 3, 5, 6, 7, 8, 9, 0}
		int fun(int b[], int n)
		{
		if (n==1)
		return b[n-1];
		else return b[n] + fun(b, n-1) + fun(b, n-1);
		}
		What is the result of the function call fun(a, 3) and What's the asymptotic running time of this function in terms of n respectively.
		A. 18, $\Theta(n)$
		B. 18, $\Theta(2^n)$
		C. 19, $\Theta(n)$
		D. 19, $\Theta(2^n)$
		E. NONE		

	MCQ	Consider the following code fragment.
		int a[] = {2, 1, 4, 3, 5, 6, 7, 8, 9, 0}
		int fun(int b[], int n)
		{
		if (n==1)
		return b[n-1];
		else return b[n] + 2*fun(b, n-1);
		}
		What is the result of the function call fun(a, 3) and What's the asymptotic running time of this function in terms of n respectively.
		A. 18, $\Theta(n)$
		B. 18, $\Theta(2^n)$
		C. 19, $\Theta(n)$
		D. 19, $\Theta(2^n)$
		E. NONE
			C	CO1

Q.No: 1(b)	**MCQ**	f(n) = 2^(2^n), g(n) =2^(n^2), h(n)=n^(n^2) Which of the following correctly represents the asymptotic relationships between the functions? (^ represents to the power) A. f(n)=O(g(n)) B. f(n)=Θ(g(n)) C. h(n)=O(g(n)) D. g(n)=Ω(f(n)) E. NONE	**E**	CO1
	MCQ	f(n) = 2^(2^n), g(n) =2^(n^2), h(n)=n^(n^2) Which of the following correctly represents the asymptotic relationships between the functions? (^ represents to the power) A. f(n)=Ω(g(n)) B. f(n)=Θ(g(n)) C. h(n)=O(g(n)) D. g(n)=Ω(f(n)) E. NONE	**A**	CO1
	MCQ	f(n) = 2^(2^n), g(n) =2^(n^2), h(n)=n^(n^2) Which of the following correctly represents the asymptotic	**C**	CO1

		relationships between the functions? (^ represents to the power) A. f(n)=O(g(n)) B. f(n)=O(h(n)) C. g(n)=O(h(n)) D. g(n)=Ω(f(n)) E. NONE		
	MCQ	f(n) = 2^(2^n), g(n) =2^(n^2), h(n)=n^(n^2) Which of the following correctly represents the asymptotic relationships between the functions? (^ represents to the power) A. f(n)=O(g(n)) B. f(n)=Θ(h(n)) C. h(n)=O(g(n)) D. h(n)=Ω(g(n)) E. NONE	**D**	CO1
Q.No: 1(c)	**MCQ**	If all the elements in an input array are same, for example {4,4,4,4,4,4}, Which of the following sorting algorithm has the lowest time complexity? A. Insertion Sort B. Quick Sort C. Merge Sort D. Both Quick & Merge Sort E. NONE	**A**	CO2
	MCQ	If all the elements in an input array are same, for example {4,4,4,4,4,4}, Which of the following sorting algorithm has the highest time complexity? A. Insertion Sort B. Quick Sort C. Merge Sort D. Both Quick & Merge Sort E. NONE	**B**	CO2
	MCQ	In an array of n integers first n/2 elements are sorted in ascending order, rest sorted in descending order. What is the minimum time required to sort the data in ascending order? A. O(log n) B. O(nlog n) C. O(n) D. $O(n^2)$ E. NONE	**C**	CO2
	MCQ	What is the minimum time required to merge two max-heaps, each having n elements, into one max heap? A. O(1) B. O(log n) C. O(nlog n) D. O(n) E. NONE	**D**	CO2
Q.No: 1(d)	**MCQ**	What will be the content of the array if 15 is inserted to an max-heap A={20, 10, 8, 6, 7, 5, 3, 3, 2}. A. {20, 15, 8, 6, 10, 5, 3, 3, 2, 7} B. {20, 15, 10, 5, 8, 6, 7, 3, 2, 3}	**A**	CO3

		C. {20, 10, 15, 5, 8, 6, 7, 3, 2, 3} D. {20, 15, 6, 8, 10, 5, 3, 3, 2, 7} E. NONE		
	MCQ	What will be the content of the array if 15 is inserted to an max-heap A={20, 8, 10, 5, 3, 6, 7, 3, 2}. A. {20, 15, 8, 6, 10, 5, 3, 3, 2, 7} B. {20, 15, 10, 5, 8, 6, 7, 3, 2, 3} C. {20, 10, 15, 5, 8, 6, 7, 3, 2, 3} D. {20, 15, 6, 8, 10, 5, 3, 3, 2, 7} E. NONE	**B**	
	MCQ	What will be the content of the array if 1 is inserted to an min-heap A={2, 3, 3, 5, 7, 6, 8, 10, 20}. A. {1, 2, 3, 5, 3, 6, 8, 10, 7, 20} B. {1, 2, 3, 3, 6, 5, 7, 10, 20, 8} C. {1, 2, 3, 5, 3, 6, 8, 10, 20, 7} D. {1, 2, 3, 6, 3, 5, 7, 10, 20, 8} E. NONE	**C**	CO3
	MCQ	What will be the content of the array if 1 is inserted to an min-heap A={2, 3, 3, 6, 8, 5, 7 10, 20}. A. {1, 2, 3, 5, 3, 6, 8, 10, 7, 20} B. {1, 2, 3, 3, 6, 5, 7, 10, 20, 8} C. {1, 2, 3, 5, 3, 6, 8, 10, 20, 7} D. {1, 2, 3, 6, 3, 5, 7, 10, 20, 8} E. NONE	**D**	CO3
Q.No: 1(e)	**MCQ**	Given items as {value, weight} pairs {{30,10},{40,20},{20,5}}. The capacity of knapsack=35. Find the maximum value output assuming items to be divisible and nondivisible respectively. A. 70, 80 B. 80, 90 C. 90, 80 D. 90, 90 E. NONE	**D**	CO3
	MCQ	Given items as {value, weight} pairs {{30,10},{40,20},{60,15}}. The capacity of knapsack=30. Find the maximum value output assuming items to be divisible and nondivisible respectively. A. 100, 100 B. 90, 100 C. 100, 90 D. 90, 90 E. NONE	**C**	CO3
	MCQ	Given items as {value, weight} pairs {{30,10},{40,20},{60,15}}. The capacity of knapsack=35. Find the maximum value output assuming items to be divisible and nondivisible respectively. A. 110, 100 B. 100, 110 C. 100, 90 D. 90, 90 E. NONE	**A**	CO3
	MCQ	Given items as {value, weight} pairs	**B**	CO3

		{{30,10},{40,20},{60,15}}. The capacity of knapsack=40. Find the maximum value output assuming items to be divisible and nondivisible respectively. A. 110, 100 B. 120, 100 C. 100, 120 D. 120, 90 E. NONE		

SECTION-B(Answer Any One Question. Each Question carries 10 Marks)

Time: 30 Minutes **(1×10=10 Marks)**

Question No	Question	CO Mapping
Q.No: 2	a) Write down the PARTITION(A, p, r) procedure with last element as pivot, where p and r are lower & upper bound of array A. If the input array is A={2, 5, 7, 9, 6, 3, 1, 8, 4}, what is the result sequence of numbers in A after making a call to PARTITION(A, 1, 9). Also show the intermediate steps of PARTITION(A, 1, 9) procedure. b) The Best case, worst case & average case time complexities are depend upon the pivot index (let it be q). Write the general recurrence for the time complexity T(n) for recursive QUICK-SORT(A, p, r) algorithm in terms of n and q. If A contains distinct elements and sorted in decreased order what will be recurrence equation and its time complexity?	CO3

	Evaluation Scheme
	● PARTITION(A, p, r) procedure : 3 Marks
	● Application of PARTITION(A, 1, 9) to the given array: 3 Marks
	● Analysis of Time Complexity of Quick Sort: 4 Marks
	● The answer of time complexity of quick sort is written only, but not derived properly, 2 or 3 marks will be deducted.
	
	Sample Solution
	a)//PARTITION procedure of QUICK-SORT algorithm with last element as pivot
	PARTITION (A, p, r)
	{
	x←A[r] //Taking last element as pivot
	i ← p-1;
	for (j←p to r-1)
	{
	if A[j] ≤ x)
	{
	i ← i+1;
	A[i] ↔ A[j]; //Internal Swap
	}
	}
	i ← i+1;
	A[i] ↔ A[r]; //Final Swap
	return i;
	}	

Application of PARTITION(A, 1, 9) to the given array A={2, 5, 7, 9, 6, 3, 1, 8, 4}

Representation of intermediate steps of pass-1

x=4 <= Last element as pivot

i=0	j=p=1	2	3	4	5	6	7	8	r=9
	2	5	7	9	6	3	1	8	$\boxed{4}$

.

0	i=j=p =1	2	3	4	5	6	7	8	r=9
	2	5	7	9	6	3	1	8	$\boxed{4}$

0	i=p=1	j=2	3	4	5	6	7	8	r=9
	2	5	7	9	6	3	1	8	$\boxed{4}$

.

0	p=1	i=2	3	4	5	j=6	7	8	r=9
	2	**5**	7	9	6	**3**	1	8	$\boxed{4}$

0	p=1	i=2	3	4	5	6	j=7	8	r=9
	2	3	7	9	6	5	1	8	$\boxed{4}$

.

0	p=1	2	i=3	4	5	6	j=7	8	r=9
	2	3	**7**	9	6	5	**1**	8	$\boxed{4}$

0	p=1	2	i=3	4	5	6	7	i=8	r=9
	2	3	1	9	6	5	7	8	$\boxed{4}$

.

0	p=1	2	3	i=4	5	6	7	8	i=r =9
	2	3	1	**9**	6	5	7	8	$\boxed{\mathbf{4}}$

As i=r so final proceeding for final swap

Pivot index=4

0	p=1	2	3	4	5	6	7	8	i=r =9
Pass-1 Result	2	3	1	$\boxed{4}$	6	5	7	8	9

LEFT PART UNSORTED LIST

RIGHT PART UNSORTED LIST

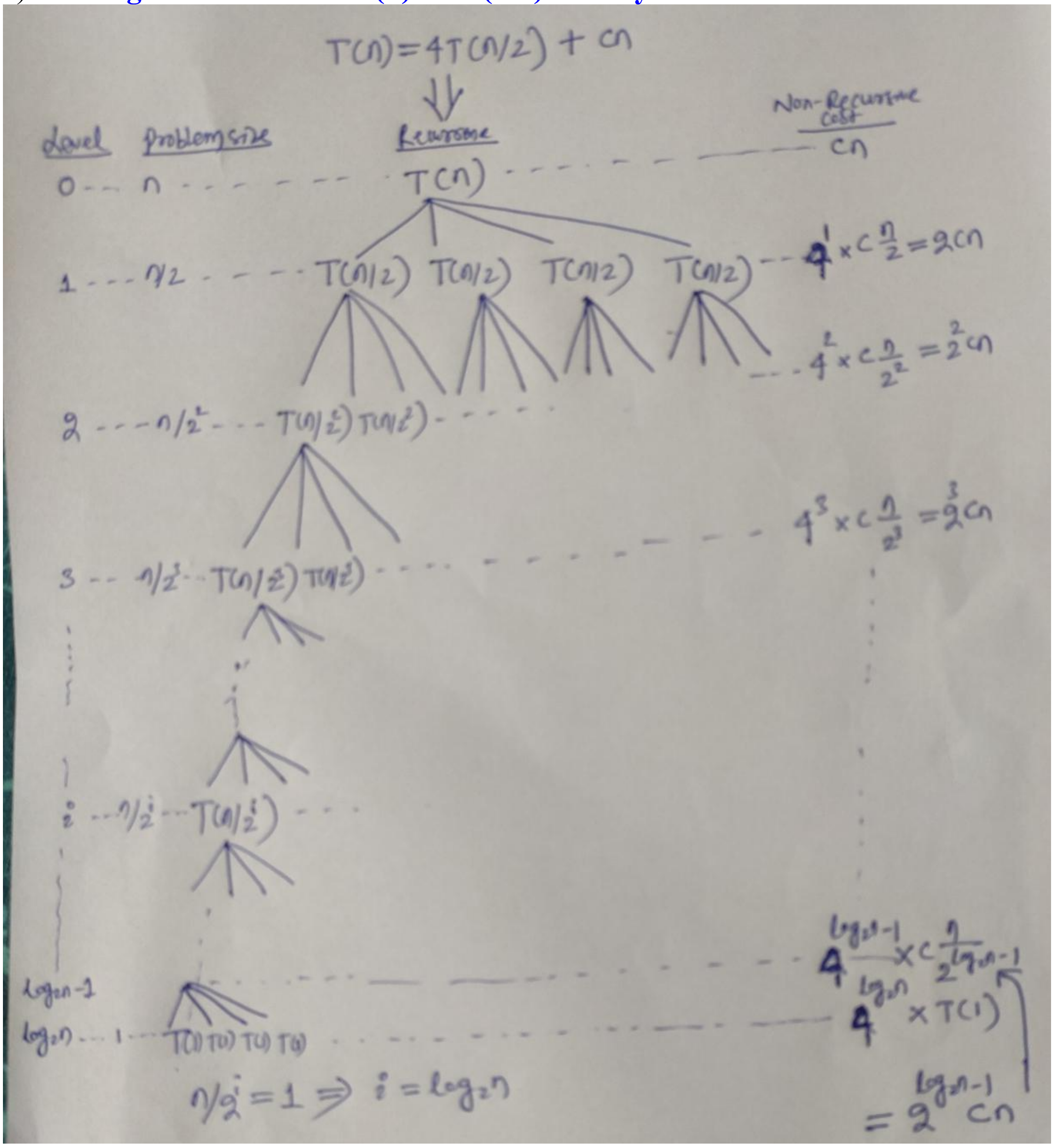
b) The general recurrence for the time complexity T(n) for recursive QUICK-SORT(A, p, r) algorithm in terms of n and q.

Referring to function call QUICK-SORT(A, 1, n), the recurrence for time complexity T(n) is derived as follows:

$$T(n) = T(n-q) + T(q-1) + n, T(1) = 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

	If A contains distinct elements and sorted in decreased order such as A={6, 5, 4, 3, 2} and pivot is chosen as last element, then PARTITION procedure will return q=1. Substituting value of q=1 in eq.-1, eq-1 becomes $T(n) = T(n-1) + n$, $T(1) = 1$ ………………………..(2) As $T(0)=1$ & $1+n \sim n$ **The solution of recurrence in eq-2, $T(n)=O(n^2)$**	
Q.No: 3	a) Draw the recurrence tree for $T(n) = 4T(n/2) + cn$, where c is a constant, and provide a tight asymptotic tight bound on its solution, verify the bound by master theorem. b) Solve the recurrence $T(n) = T(n-1) + 2^n$ using master method by changing variable first to transfer the recurrence to an appropriate form or solve by any other method.	CO1
	Evaluation Scheme ● Representation of recurrent tree for given recurrence: 3 Marks ● Verification of the solution found by recurrence tree method by master theorem: 2 Marks ● Solving the recurrence $T(n) = T(n-1) + 2^n$ correctly: 5 marks **Sample Solution** **b) Solving the recurrence $T(n) = T(n-1) + 2^n$ by change of variable with master theorem** **$T(n) = T(n-1) + 2^n$ ……………… (1)** Let $T(n) = S(2^n) \Rightarrow T(n-1) = S(2^{n-1})$. Substitute these values in eq-1. $S(2^n) = S(2^{n-1}) + 2^n$ ------------------(2) Now assume $m=2^n$ and substitute this value in eq-2, the recurrence becomes $S(m) = S(m/2) + m$ ……………….(3) Eq-3 is replicates the master theorem form. a=1, b=2, k=1, p=0 $b^k=2^1=2$, $a< b^k$ and $p\geq 0$ so case-3.a of master theorem is applicable. As per case 3.1, the solution to the recurrence is $S(m) = \Theta(m^k \log^p m) = \Theta(m^1 \log^0 m) = \Theta(m)$ $\Rightarrow S(2^n) = \Theta(2^n)$ $\Rightarrow$ **$T(n) = \Theta(2^n)$ (Answer)**	

a) **Solving the recurrence T(n) = 4T(n/2) + cn by recurrence tree method**



Now,

Time Complexity

= Sum of non-recursive cost occurred from level 0 to last level-1 + Total cost at last level

= (cn + 2cn + 2^2cn + 2^3cn + ….. +$2^{\log (n-1)}$) + $4^{\log n}$ x T(1)

= cn{20 + 21 + 22 +23 +……..+$2^{\log (n-1)}$} + $n^{\log 4}$ x 1

$= cn\left\{\frac{2^{\log_2 n-1+1}-1}{2-1}\right\} + n^{\log_2 2^2}$ = cn($2^{\log n}$) +$n^{2\log 2}$ = cn($n^{\log 2}$) +n^2 =cn^2 + n^2 = $\Theta(n^2)$

Solving the recurrence T(n) = 4T(n/2) + cn by master theorem

a=4, b=2, k=1, p=0

Now $b^k=2^1=2$, as a> b^k => T(n) = $\Theta(n^{\log_2 4})$ = $\Theta(n^2)$

Q.No: 4	a) Write the MERGE-SORT(A, p, r) procedure where at each step it divides the the array/sub-array into two parts such that second part contains elements twice of first part instead of dividing at middle. b) For array A={10, 45, 15, 40, 10, 20, 40, 25, 35}, MERGE-SORT(A, 9) is	CO3

applied to sort the array in ascending order. Show in diagram how this procedure is applied to this array.

Evaluation Scheme

- MERGE-SORT(A, p, r) procedure : 5 Marks
- Application of MERGE-SORT(A, 9) to the given array with the division taking q=(p+r)/2 or q⟵(r+2p-2)/3 : 5 Marks
- A function is used but not defined by the students, 1 mark will be deducted. Example: The name MERGE procedure is used in MERGE-SORT procedure but is not defined anywhere.

Sample Solution

a) MERGE-SORT procedure

```
//merge sort is applied to the array A to sort the array in ascending order from the
//lower bound/index p to upper bound/index r.
MERGE-SORT (A, p, r)
{
    if (p<r)
    {
        //divides the the array/sub-array into two parts such that second part
        //contains elements twice of first part
        q⟵(r+2p-2)/3;
        MERGE-SORT (A, p, q);
        MERGE-SORT (A, q+1, r);
        MERGE (A, p, q, r);
    }
}
```

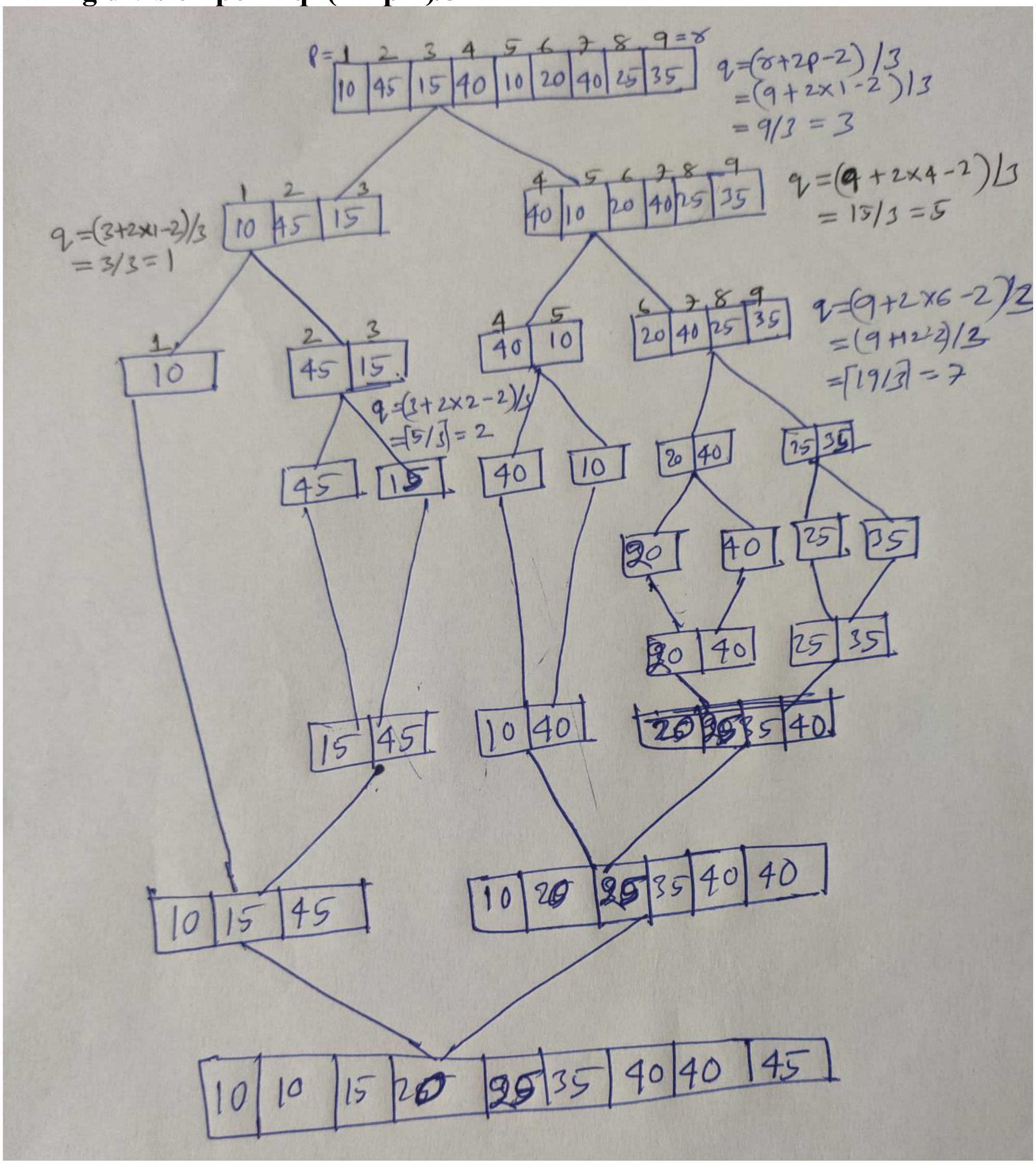
MERGE Procedure

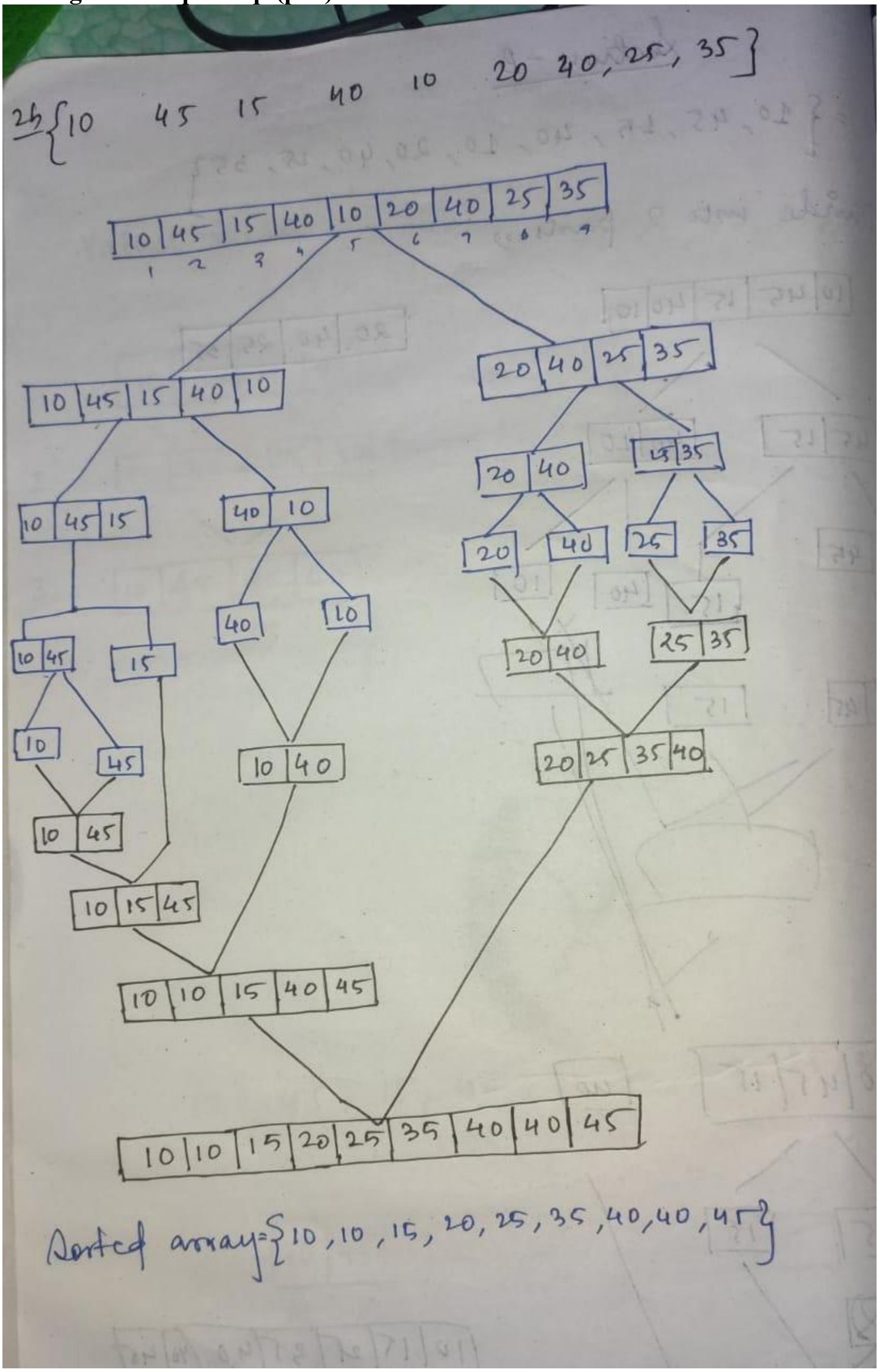
```
//Merge procedure to merge/combine the elements already sorted in array A,
//from index p to r and index q+1 to r into a sorted array.
MERGE(A, p, q, r)
{
    n1 ← q − p + 1
    n2 ← r − q
    //Create arrays L[1 . . n1 + 1] and R[1 . . n2 + 1]
    for i ← 1 to n1
        L[i] ← A[p + i − 1]
    for j ← 1 to n2
        R[j] ← A[q + j ]
    L[n1 + 1] ← ∞
    R[n2 + 1] ← ∞
    i ← 1
    j ← 1
    for k ← p to r
    {
        if L[i ] ≤ R[ j]
        {
            A[k] ← L[i]
            i ← i + 1
        }
        else
        {
            A[k] ← R[j]
```

```
            j ← j + 1
        }
    }
}
```

b) Application of MERGE-SORT (A, 1, 9) to the array A={10, 45, 15, 40, 10, 20, 40, 25, 35}

Taking division point q=(r+2p-2)/3



	Taking division point q=(p+r)/2 	
Q.No: 5	a) Write an algorithm MAX-HEAP-CHANGE(A n, i, key) that rebuilds the n-element max-heap if the value at index i is changed to the value as key. b) Apply this algorithm to the max-heap A={20, 15, 18, 10, 8, 12, 9, 6, 4, 8, 10} if at index 5 the value is changed to 25 and then at index 2 of modified heap	CO3

	the value is changed to 3. Assume root is at index 1. Show the process & result in max-heap diagram.	

Evaluation Scheme

- MAX-HEAP-CHANGE(A n, i, key) Algorithm : 5 Marks
- Application of the above algorithm to the max-heap A={20, 15, 18, 10, 8, 12, 9, 6, 4, 8, 10}. There is a typo error in last element 10. The correct value is 1 in place of 10. Identifying the given array is not a max-heap and applying the above algorithm after converting the array into a max-heap: 5 marks
- A function is used but not defined by the students, 1 mark will be deducted. Example: The MAX-HEAPIFY is used, but not defined.

Sample Solution

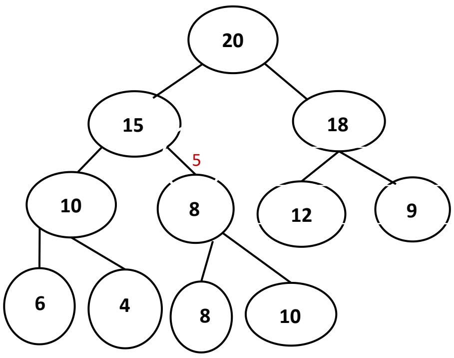
Method-1 : By using known Coreman book algorithms	**Method-2 :**

MAX-HEAP-CHANGE(A n, i, key)
{
if (key<A[i])
{
A[i] ← key;
MAX-HEAPIFY(A, n, i)
}
else if(key>A[i])
HEAP-INCREASE-KEY (A, n, i, key)
}
 	
Where,
HEAP-INCREASE-KEY (A, n, i, key)
{
if (key < A[i])
{
Print "error-new key is smaller than current key";
Exit;
}
A[i] ← key;
while (i>1 and A[PARENT(i)] < A[i])
{
A[i] ↔ A[PARENT(i)];
i ← PARENT(i);
}
}
	MAX-HEAP-CHANGE(A n, i, key)
	{
	
	if (key<A[i])
	{
	A[i] ← key;
	MAX-HEAPIFY(A, n, i)
	}
	else if(key>A[i])
	{
	A[i] ← key;
	MAX-HEAPIFY-UP(A, n, i)
	}
	}
	Where,
	/*MAX-HEAPIFY-UP rearranges the nodes from index i max-possiblely to root to satisfy the max heap property.*/
	
	MAX-HEAPIFY-UP(A, n, i)
	{
	while (i>1 and A[PARENT(i)] < A[i])
	{
	A[i] ↔ A[PARENT(i)];
	i ← PARENT(i);
	}
	}

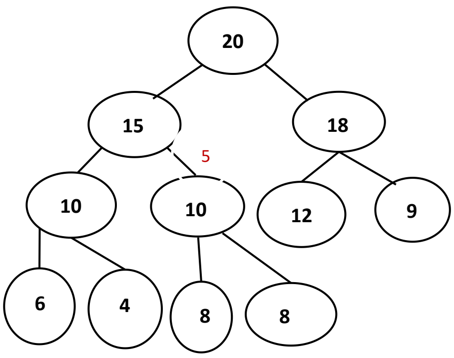
```
/*Max-Heapify : Given a tree that is a max-heap of n-element array,
except for node i, Max-Heapify function arranges node i and it's subtrees to
satisfy the heap property.*/

MAX-HEAPIFY(A, n, i)
{
     l ← LEFT(i);
     r ← RIGHT(i);
     if   (l ≤ n and A[l] > A[i])
            largest = l;
     else
            largest = i;
     if   (r ≤ n and A[r] > A[largest])
            largest = r;
   if (largest != i)
   {
            A[i] ↔ A[largest]; // swaping
            MAX-HEAPIFY(A, n, largest);
     }
}
```

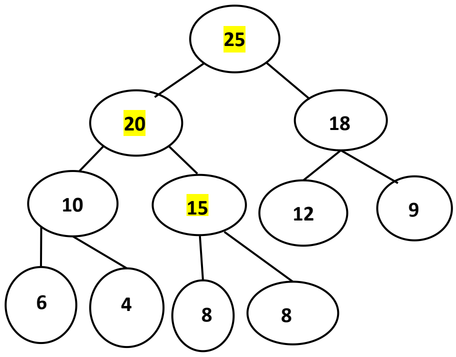
b) **The given Max Heap corresponds to array A={20, 15, 18, 10, 8, 12, 9, 6, 4, 8, 10} is as follows.**



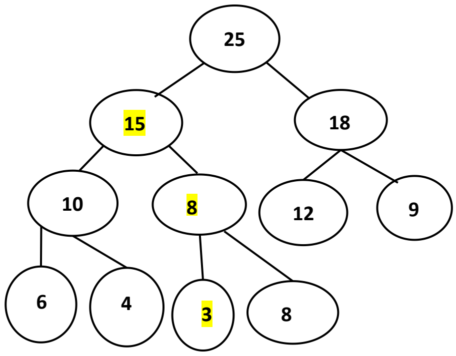
But value at index 5 does not satisfy the max-heap property. So applying MAX-HEAPIFY(A, 11, 5), the given heap is converted into max-heap as follows.



if at index 5 the value is changed to 25, the above max-heap becomes



if at index 2 the value is changed to 3, the above max-heap becomes



This is the final max-heap A={25, 15, 18, 10, 8, 12, 9, 6, 4, 3, 8}

Q.No: 6

Prof. HariBol holds shares on some commodities in his Demat Account as given in the Table below. Write an algorithm for Prof. HariBol to decide number of shares to be sold from different commodities to make maximum profits subject to generate fixed amount of money. Find the number of shares to be sold by Prof. HariBol for each commodity to maximize the profits subject to generate a liquid cash of Rs**800**/-. Fraction of a share can be sold.

Table: Prof. HariBol holdings of shares (Assume N=Your Roll Number)

Commodity Name	Unit Selling Price in Rs (S)	Profits per Share in Rs (P)	No. of Shares
Gold	90	9	5
Silver	40	5	8
Crude Oil	80	20	N MOD 5
Sugar	55	5	4
Wheat	30	3	5
Rubber	30	5	3
Mentho Oil	45	15	6
Natural Gas	21	3	2
Cotton	10	2	20

Evaluation Scheme

- Algorithm for Prof. HariBol to decide number of shares to be sold from different commodities to make maximum profits subject to generate fixed amount of money : 5 Marks
- Application of the algorithm to the given problem: 5 Marks

Answer

N MOD 5	Gold	Silver	Crude Oil	Sugar	Wheat	Rubber	Mentho Oil	Natural Gas	Cotton	No. Of Shares to be Sold
0	0	4.95	0	0	0	3	6	2	20	<=
1	0	2.95	1	0	0	3	6	2	20	<=
2	0	0.95	2	0	0	3	6	2	20	<=
3	0	0	3	0	0	3	6	0	20	<=
4	0	0	4	0	0	0.33	6	0	20	<=

Sample Solution

a) This problem is similar to fractional Knapsack problem.

```
GREEDY-SHARETRADING(S, P, X, U, n)
//S selling price per share; P profit per share; X solution vector;
//N number of each share; n types of share; U total liquid cash
//Arrange the commodities in increasing order of S/P
{
    for i =   1 to n do
        X[i] = 0    // Initilize x
    for i = 1 to n do
    {
        if (S[i]xN[i] > U) then break;
        else
        {
             X[i] = N[i]
             U = U – S[i]xN[i]
        }
    }
    if (i ≤ n) the X[i] = U/S[i]
}
```

b) Application of the GREEDY-SHARETRADING(S, P, X, U, n) algorithm to the given example

Step-1: Calculate S/P for each commodities

Commodity Name	Selling Price per Share in Rs (S)	Profits per Share in Rs (P)	S/P	No. of Shares
Gold	90	9	10	5
Silver	40	5	8	8
Crude Oil	80	**20**	4	**N MOD 5**
Sugar	55	5	11	4
Wheat	30	3	10	5
Rubber	30	5	6	3

Mentho Oil	45	15	3	6
Natural Gas	21	3	7	2
Cotton	10	2	5	20

N.B: Calculation of P/S is also correct.

Step-2: Arrange the commodities in increasing order of S/P. So the table will become as follows:

Taking N=1905127, N MOD 5 = 2

Commodity Name	**Unit Selling Price in Rs (S)**	**Profits per Share in Rs (P)**	**No. of Shares (N)**	**P/S**	**REMARKS U=800**	**No of Shares to be sold (X)**
Mentho Oil	45	15	6	3	45*6≤800 yes, so total share can be sold. Rest Money Required=U=800-45*6=530	**6**
Crude Oil	80	**20**	**2**	4	80*2≤530 yes. So total shares can be sold. Rest Money Required=U=530-80*2=370	**2**
Cotton	10	2	20	5	10*20≤370 yes. So total shares can be sold. Rest Money Required=U=370-10*20=170	**20**
Rubber	30	5	3	6	30*3≤170 yes. so total shares can be sold. Rest Money Required=U=170-30*3=80	**3**
Natural Gas	21	3	2	7	21*2≤80 yes. so total shares can be sold. Rest Money Required=U=80-21*2=38	**2**
Silver	40	5	8	8	40*8≤38 no. so part of the shares can be	**0.95**

					sold. i.e 38/40=>0.95	
Gold	90	9	5	10		**0**
Wheat	30	3	5	10		**0**
Sugar	55	5	4	11		**0**

Total Liquid Cash = 6x45 + 2x80 + 20x10 + 3x30 + 2x21 + 0.95x40 = Rs800/-
Total profit earned = 6x15 + 2x20 + 20x2 + 3x5 + 2x3 + 0.95x5 = Rs.195.75/-

Taking N=2005238, N MOD 5 = 3

Step-1: Same
Step-2: Arrange the commodities in increasing order of S/P. So the table will become as follows:

Commodity Name	**Unit Selling Price in Rs (S)**	**Profits per Share in Rs (P)**	**S/P (increasing order)**	**No. of Share (N)**	**No of Shares to be sold (X)**
Mentho Oil	45	15	3	6	6
Crude Oil	80	20	4	**3**	**3**
Cotton	10	2	5	20	20
Rubber	30	5	6	3	3
Natural Gas	21	3	7	2	0
Silver	40	5	8	8	0
Wheat	30	3	10	5	0
Gold	90	9	10	5	0
Sugar	55	5	11	4	0

Total Liquid Cash = 6x45 + 3x80 + 20x10 + 3x30 = Rs800/-
Total profit earned = 6x15 +3x20 + 20x2 +3x5 = Rs205/

Q.No: 7

Prof. BolHari holds shares on some commodities in his Demat Account as given in the Table below. Find the number of shares to be sold by Prof. BolHari for each commodity to maximize the profits subject to generate a liquid cash of Rs**700**/-. Fraction of a share can be sold. Write an algorithm for Prof. BolHari to decide number of shares to be sold from different commodities to make maximum profits subject to generate fixed amount of money.

Table: Prof. BolHari holdings of shares (Assume N=Your Roll Number)

Commodity Name	Unit Selling Price in Rs (S)	Profits per Share in Rs (P)	No. of Shares
Gold	80	8	5
Corn	40	5	N MOD 5

Crude Oil	80	20	3
Soyabeans	55	5	4
Wheat	30	3	4
Rubber	30	5	6
Mentho Oil	45	15	5
Natural Gas	21	3	2
Copper	10	2	15

Evaluation Scheme

- Algorithm for Prof. BolHari to decide number of shares to be sold from different commodities to make maximum profits subject to generate fixed amount of money : 5 Marks
- Application of the algorithm to the given problem: 5 Marks

Answer

N MOD 5	Gold	Corn	Crude Oil	Soyabeans	Wheat	Rubber	Mentho Oil	Natural Gas	Copper	No. Of Shares to be Sold
0	0	0	3	0	0	2.83	5	0	15	<=
1	0	1	3	0	0	2.83	5	0	15	<=
2	0	2	3	0	0	2.83	5	0	15	<=
3	0	3	3	0	0	2.83	5	0	15	<=
4	0	3	3	0	0	2.83	5	0	15	<=

Same Answer in all cases.

Sample Solution

a) This problem is similar to fractional Knapsack problem.

```
GREEDY-SHARETRADING(S, P, X, U, n)
//S selling price per share; P profit per share; X solution vector;
//N number of each share; n types of share; U total liquid cash
//Arrange the commodities in increasing order of S/P
{
    for i =   1 to n do
        X[i] = 0    // Initilize x
    for i = 1 to n do
    {
        if (S[i]xN[i] > U) then break;
        else
        {
            X[i] = N[i]
            U = U – S[i]xN[i]
        }
    }
    if (i ≤ n) the X[i] = U/S[i]
}
```

b) Application of the GREEDY-SHARETRADING(S, P, X, U, n) algorithm to the given example

Step-1: Calculate the S/P of each commodities

Commodity Name	Selling Price per Share in Rs (S)	Profits per Share in Rs (P)	S/P	No. of Shares (N)
Gold	80	8	10	5
Corn	40	**5**	8	**2**
Crude Oil	80	20	4	3
Soyabeans	55	5	11	4
Wheat	30	3	10	4
Rubber	30	5	6	6
Mentho Oil	45	15	3	5
Natural Gas	21	3	7	2
Copper	10	2	5	15

Step-2: Arrange the commodities in increasing order of S/P. So the table will become as follows:

Taking N=1905127, N MOD 5 = 2

Commodity Name	Unit Selling Price in Rs (S)	Profits per Share in Rs (P)	No. of Shares (N)	S/P	REMARKS U=700	No of Shares to be sold (X)
Mentho Oil	45	15	5	3	45*5≤700 yes, so total share can be sold. Rest Money Required=U=700-45*5=475	5
Crude Oil	80	20	3	4	80*3≤475 yes, so total share can be sold. Rest Money Required=U=475-80*3=235	3
Copper	10	2	15	5	10*15≤235 yes, so total share can be sold. Rest Money Required=U=235-10*15=85	2

Rubber	30	5	6	6	30*6≤85 NO, **so part of the shares can be sold. i.e 85/30=>2.83**	2.83
Natural Gas	21	3	2	7		0
Corn	40	**5**	**2**	8		0
Gold	80	8	5	10		0
Wheat	30	3	4	10		0
Soyabeans	55	5	4	11		0

Total Liquid Cash = 5x45 + 3x80 + 15x10 + 2.83x30 = Rs700/-
Total profit earned = 5x15 +3x20 + 20x2 + 2.83x5 = Rs189.15