Write the number of the question in this column.

1. a) An algorithm is a way of solving a problem. It contains step(s) which are necessary to achieve a particular task. Examples of representing an algorithm are flowcharts and pseudocodes.

b) Asymptotic analysis states that the exact memory and time required for an algorithm isn't important but rather we identify how they change with increasing problem size (ratio). Ex: In one platform an algorithm has $O(n/2)$ complexity while in another platform it has $O(n/4)$ complexity but ultimately both have $O(n)$ complexity.

c)
i. = $O(n^3)$
ii. = $O(n^2)$
iii. = $O(n \log(n))$
iv. = $O(n)$
v. = $O(n)$

d)
* The if statement in the function has $O(1)$ complexity.
* The function will call itself $(n-1)$ no of times.
* Final complexity = $O(n-1) * O(1)$
= $O(n)$.

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

2. a. int bsearch (int key, int size, int array[])

b.
1. // Initialize first to 0.
2. // Initialize last to ~~siz~~ value of size-1.
3. // Initialize found to 0.
4. // Initialize position to -1.
6. // Set while loop condition.
7. // Calculate the value of middle.
8. // Set found to 1 if key is found.
9. // Set position to middle if key is found.
10. // Set last to value of middle-1 if key is less than array[middle].
11. // Else, set first to value of middle+1.
12. // Close the loop.
13. // Return the value of position and terminate the function.

c.

Variable	Initially	After Iteration 1	After Iteration 2	After Iteration 3
key	23	23	23	23
size	15	15	15	15
first	0	8	8	8
last	14	14	10	8
found	0	0	0	0
position	-1	-1	-1	-1
(!found &&	true	true	true	true
first<=last)	~~NA~~			
middle	NA	7	11	9
array[middle]	NA	19	30	25

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

3. a.

```
int minIndex (float d[], int size) {
    int minIn = 0, i = 1;
    float min = d[0];
    for ( ; i < size ; i++)
    {
        if (d[i] < min)
        {   min = d[i];
            minIn = i;
        }
    }
    return minIn; }

void swap (float *p1, float *p2) {
    float temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp; }

void selectionSort (float d[], int size) {
    if (size == 1) return;
    if (size > 1) {
    swap (&d[minIndex (d,size)], d);
    return selectionSort (&d[1], size-1); } }
```

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

b. * In the minIndex function, statements outside ~~the~~ for loop have O(1) complexity.

* and inside the

* Even though the for loop runs for decreasing sizes, the complexity of it is still O(n).

* Final complexity = O(n) * O(1) + O(1).

* ~~The swap function has~~ = O(n).

2

* All statements in swap function have O(1) complexity.

* So, final complexity = ~~O(n)~~. O(1).

2

* In the selectionSort function the swap function has O(1) complexity as already mentioned, but since it calls

* ~~Though the function calls itself for decreasing sizes~~ the minIndex function, complexity is O(n).

* The selectionSort function calls itself (n-1) ~~times~~ times though it is for decreasing sizes, ultimately complexity is O(n) here.

4

Thus final complexity is = ~~O(n-1) *~~

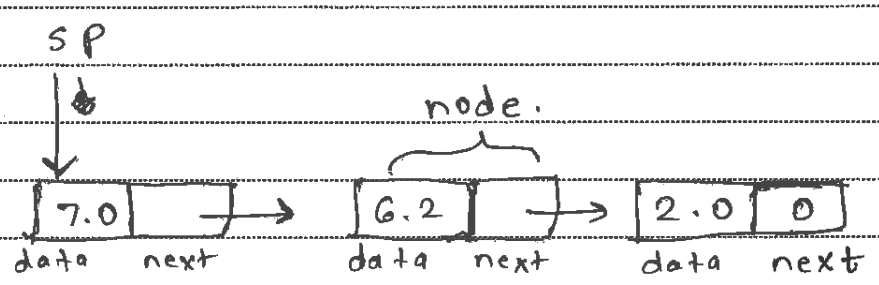
O(n) * O(n) = O(n²).

20

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

4. a. ~~stack~~



b.

```
struct node * makenode (float item) {
    struct node *p = (struct node*) malloc(sizeof(
struct node * p = (struct node*) malloc (sizeof ( struct node));
    if ( !p)  return 0;
    p -> data = item;
    p -> next = 0;
    return p;  }

void init ( struct stack * s) {
    s -> sp = 0;  }

int full ( struct stack * s) {
    return 1;  }

int empty (struct stack * s) {
    return s -> sp == 0;  }
```

06

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

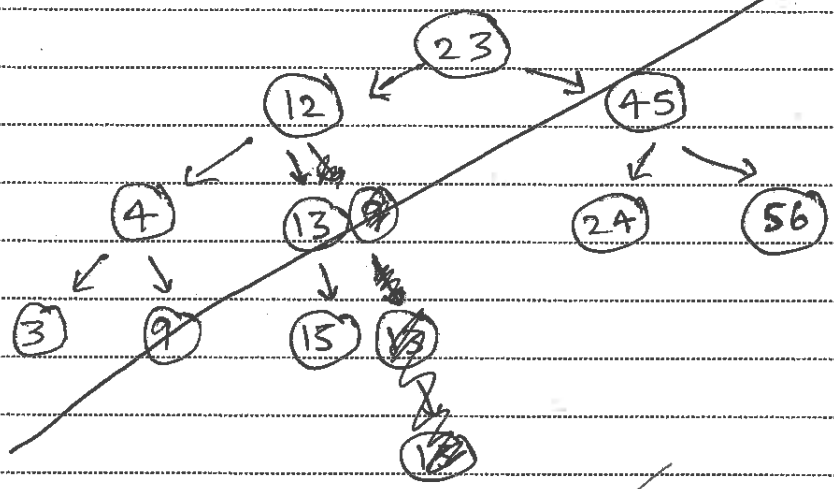
```
int push ( struct stack *s, float item ) {
    struct node *p = makenode (item);
    if (!p)  return 0;
    if (empty(s))  s -> sp = p;
    else
    { p -> next = s -> sp;
      s -> sp = p;  }
    return 1;  }

float pop ( struct stack *s) {
    ~~if (empty(s))~~
    float temp = s -> sp -> data;
    struct node *p = s -> sp;
    s -> sp = s -> sp -> next;
    free (p);
    return temp;  }

float top ( struct stack *s) {
    return s -> sp -> data;  }
```

18

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

5. a.

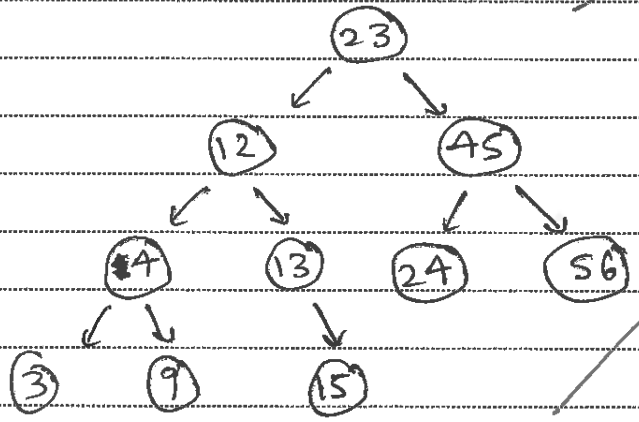


b. i. 23, 12, 4, 3, 9, 13, 15, 45, 24, 56.

ii. 3, 4, 9, 12, 13, 15, 23, 24, 45, 56.

iii. 3, 9, 4, 15, 13, 12, 24, 56, 45, 23.

a.



c.
```
struct node { int data;
              struct node *left, *right; };
```

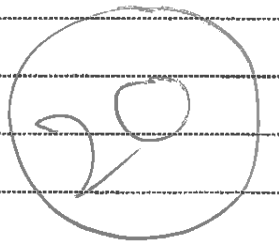
08.

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

d)
```
void preorder (struct node *tree) {
    if (!tree) return 0;
    printf ("%i", tree -> data);
    printf ("%i \t", tree -> data);
return preorder (tree -> left);
return preorder (tree -> right); }
```

e)
```
struct node * find (struct node * tree, int key){
    struct node * temp;
    if (!tree) return 0;
  if (tree -> data == key) return tree;
  if (key < tree -> data) temp= find (tree -> left, key);
  if (!temp) temp= find (tree -> right, key);
    return temp; }
```



Write the number of the question in this column.

Write the number of the question in this column.

NSBM GREEN UNIVERSITY TOWN

Write the number of the question in this column.

National School of Business Management
BSc in Management Information Systems (UGC) -16.1
BSc in Software Engineering (UGC) -16.1
BSc in Software Engineering /Computer Networks / Security (Ply) 16.1
BSc in Computer Science – 16.1
1st Year 2nd Semester Examination
Algorithms and Data structures - CS106.3

Time: 03Hrs
Date: 29th Mar 2017

Answer all Questions.

Question 1 – 20 Marks

(a) Briefly explain what an algorithm is in the context of Computing. [5 Marks]

(b) Briefly explain, giving an example, how asymptotic analysis can isolate the algorithm efficiency from the machine and platform dependency. [5 Marks]

(c) Simplify the following Big-O expressions [5 Marks]

i. $O(2n^3+5n-10)$
ii. $O(2n^2+10^2n)+O(n)$
iii. $O(n)*O(\log(n))$
iv. $O(n)+O(\log(n))$
v. $n*O(1)$

(d) Giving reasons, evaluate the time complexity of the following function. [5 Marks]

```
int fact(int n)
{   if(n==1) return 1;
    return n*fact(n-1);   }
```

Question 2 - 20 Marks

Following code segment implements the binary search algorithm.

```
1   first = 0;
2   last = size -1;
3   found = 0;
4   position = -1;
6   while(!found && first <=last) {
7   middle = (first+last)/2;
8   if(array[middle] == key) {   found = 1;
9                                position = middle;  }
10  else if(key<array[middle]) last = middle – 1;
11  else first = middle + 1;
12  }
13  return position;
```

(a) If the above code segment to write inside a function called bsearch() what will be the return type and required arguments for the function? Give your answer by writing the function header including return data type and argument declarations. [5 marks]

(b) Write down a comment line you would include in the above code against each line to illustrate the function of each line or statement. You do not have to copy the code - just put the line number and your comment in your answer script. [5marks]

(c) Copy the following table into your answer script and complete it for each iteration for the problem scenario given below to carry out a desk-check of the code given above.

Variable	initially	After iteration 1	After iteration 2	After iteration 3
key	23			
size	15			
first	0			
last	14			
found	0			
position	-1			
(!found && first <=last)	true			
middle	NA			
array[middle]	NA			

array[]

2	5	9	10	12	15	18	19	23	25	29	30	35	43	45
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

key =23 [10 Marks]

Question 3 - 20 Marks

The following is a skeleton of a selection sort implementation in C.

```
int minIndex(float d[], int size){          // return the index of
        ...                                     // the min in the given array
}
void swap(float *p1, float* p2) {           // swap two vars
        ...
}
void selectionSort(float d[], int size){
        ...
}
```

```
swap ( minIn(), &d[0])
return selectionsort (&d[1], size-1);
```

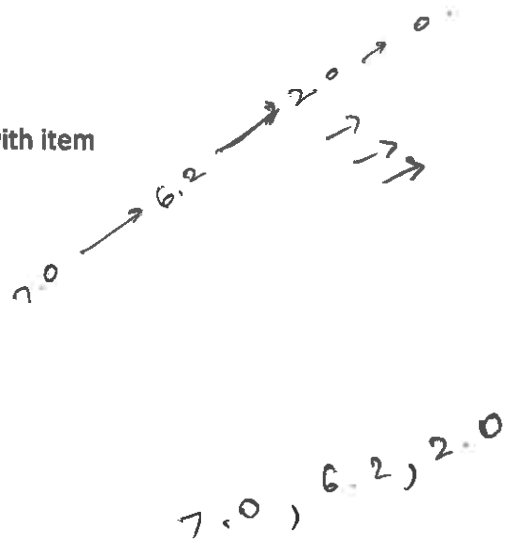
(a) Write C code to implement the above selection sort algorithm. [10 Marks]

(b) Evaluate step by step, giving reasons, the time complexity of each of the above functions in terms of the Big-O notation. [10 Marks]

Question 4 - 20 Marks

Following code intends to implement a dynamic stack.

```
struct node{    float data;
                        struct node* next;     };
struct stack{   struct node* sp;        };

struct node* makenode(float item){  // make a new node with item
...
}
void init(struct stack * s){...}  // initialize sp
int full(struct stack * s){...}             // return 1 if full
int empty(struct stack * s){...}            // return 1 if empty

int push(struct stack *s, float item){ ...
}
float pop(struct stack *s){ ...
}
float top(struct stack *s){...}
```



(a) Write a clear diagram to show the status of the stack structure instance, nodes, stored values and node linking after pushing the values 2.0, 6.2 and 7.0. [6 marks]

(b) Write code for each function above to complete the stack implementation. [14 Marks]

Question 5 - 20 Marks

(a) Draw a binary search tree generated by inserting the following items in the given order.

23, 45, 12, 4, 56, 9, 13, 15, 24, 3 [4 Marks]

(b) Draw the sequence of items you process, if the BST is traversed by,

i. pre-order, (VLR)

ii. in-order, (LVR)

iii. post-order, (LRV) tree walking methods. [6 marks]

leaf break

(c) Write down a node structure in C, suitable to implement the above BST. [2 marks]

(d) Write a C function to display the above BST in **pre-order** traversal. [4 Marks]

(e) Write a C function to find a value (key) in the BST by traversing the BST in pre-order manner. [4 Marks]

VLR)

*****End of the paper*****