0212

National University of Computer and Emerging Sciences

FAST School of Computing Fall-2022 Islamabad Campus

CS-2001: Data Structures

Serial No:

Sessional Exam-I

Total Time: 1 Hour

Total Marks: 60

Saturday, 26th September, 2022

Course Instructors

Marium Hida, Owais Idrees, Adnan Tariq

Signature of Invigilator

Student Name	Roll No.	Section	Signature
Nouman Amjad	21i-0853	E	Numj

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **fourteen (14)** different printed pages including this title page. There are a total of 3 questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Total
Marks Obtained	11	20	07	38
Total Marks	20	20	20	

Good

Question 1 [20 Marks]

1. Consider the following code snippet of pancake sorting algorithm.

```
void flip(int arr[], int i)
{
    int temp, start = 0;
    while (start < i) {
        temp = arr[start];
        arr[start] = arr[i];
        arr[i] = temp;
        start++;
        i--;
    }
}
int findMax(int arr[], int n)
{
    int mi, i;
    for (mi = 0, i = 0; i < n; ++i)
        if (arr[i] > arr[mi])
            mi = i;
    return mi;
}
void pancakeSort(int* arr, int n)
{
    for (int curr_size = n; curr_size > 1; --curr_size)
    {
        int mi = findMax(arr, curr_size);

        if (mi != curr_size - 1) {
            flip(arr, mi);  // Flip Up
            flip(arr, curr_size - 1); // Flip Down
        }
    }
}
int main()
{
    int arr[] = { 1, 4, 5, 2, 3, 8, 6, 7, 9, 0 };
    int n = sizeof(arr) / sizeof(arr[0]);
    pancakeSort(arr, n);
    return 0;
}
```

a) Perform dry run and write the output of each iteration after flip up and flip down in the table below. [8 marks]

curr_size	mi	arr after flip(arr,mi)	arr after flip(arr, curr_size-1)
10	8	9, 7, 6, 8, 3, 2, 5, 4, 1, 0	0, 1, 4, 5, 2, 3, 8, 6, 7, 9
9	6	8, 3, 2, 5, 4, 1, 0, 6, 7, 9	7, 6, 0, 1, 4, 5, 2, 3, 8, 9
8	0	7, 6, 0, 1, 4, 5, 2, 3, 8, 9	3, 2, 5, 4, 1, 0, 6, 7, 8, 9
7	6	~~6, 3, 0, 6, 1, 4, 5, 2, 8, 7, 9~~ same	3, 2, 5, 4, 1, 0, 6, 7, 8, 9
6	2 ✓	5, 2, 3, 4, 1, 0, 6, 7, 8, 9	0, 1, 4, 3, 2, 5, 6, 7, 8, 9
5	2 ✓	4, 1, 0, 3, 2, 5, 6, 7, 8, 9	2, 3, 0, 1, 4, 5, 6, 7, 8, 9
4	1 ✓	3, 2, 0, 1, 4, 5, 6, 7, 8, 9	1, 0, 2, 3, 4, 5, 6, 7, 8, 9
3	2 ✓	1, 0, 2, 3, 4, 5, 6, 7, 8, 9	1, 0, 2, 3, 4, 5, 6, 7, 8, 9
2	0 ✓	1, 0, 2, 3, 4, 5, 6, 7, 8, 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9

07

11

b) What is the worst-case time complexity of the above algorithm? Justify your answer by providing the time complexity of flip and findMax operations. [2 marks]

FindMax = $\frac{n(n+1)}{2}$

02

2. Consider the following code snippet.

```
void CocktailSort(int a[], int n)
{
    bool swapped = true;
    int start = 0;
    int end = n - 1;

    while (swapped) {
        swapped = false;
        for (int i = start; i < end; ++i) { //Forward pass
            if (a[i] > a[i + 1]) {
                swap(a[i], a[i + 1]);
                swapped = true;
            }
        }
        if (!swapped)
            break;
        swapped = false;
        --end;
        for (int i = end - 1; i >= start; --i) { //backward pass
            if (a[i] > a[i + 1]) {
                swap(a[i], a[i + 1]);
                swapped = true;
            }
        }
        ++start;
    }
}
int main()
{
    int a[] = { 5, 1, 4, 2, 8, 0, 2 };
    int n = sizeof(a) / sizeof(a[0]);
    CocktailSort(a, n);
    return 0;
}
```

a) Perform dry and write the output after each forward and backward pass in the table below. [5 marks]

Pass (Forward/backward)	Starting value of i	a (Elements in array after the pass)
Forward	0	1, 4, 2, 5, 0, 2, 8
backward	~~4~~ 5	0, 1, 4, 2, 5, 2, 8
forward	1	0, 1, 2, 4, ~~5~~ 2, 5, 8
Backward	~~3~~ 4	0, 1, 2, 4, 2, 5, 8
Forward	2	0, 1, 2, 2, 4, 5, 8
backward	2	0, 1, 2, 2, 4, 5, 8
		loop breaked (No swap.

b) What is the best-case time complexity of the above algorithm? Justify your answer! [2 marks]

best case = O(1)

& best case is when array is already sorted.

c) What is the worst-case time complexity of the above algorithm? Justify your answer! [1 mark]

$n(\frac{n}{2}+\frac{n}{2}) = n^2$

$O(n^2)$

worst case is when outer loop runs n times and inner loops run $\frac{n}{2}$ times.

d) What problem in the Bubble-Sort does the Cocktail-Sort tend to solve to improve the performance? [2 marks]

because it is sorting in both direction (forward and backward)

National University of Computer and Emerging Sciences

FAST School of Computing Fall-2022 Islamabad Campus

Question 2 [20 Marks]

1. You are given a multidimensional array containing information regarding final exam scores of top students of fast from different campuses. Your array is given as

int Campus=5, school=3, students=10, final_scores=6;
double score [campus][school][students][final_scores]
5 3 10 6

a) Given the base Address as 1000, find the address of score [2][1][7][4]. [7 Marks]

base = 1000

Address of score [2][1][7][4] =

base + 2(3 x (10x(6x8)) + 1(10(6x8)

+ 7(6x8) + (4 x 8)

= 1000 + 2880 + 480 + 336 + 32

= ~~1000~~ 4728.

If base Address is 1000 then score[2][1][7][

will be present at 4728.

2. Consider the following code snippet: 9

```
void foo(int A[], int n)
{
    if (n < 1) return;
    int write_index = n - 1; 8
    int read_index = n - 1; 8

    while(read_index >= 0)
    {
        if(A[read_index] != 0)
        {
            A[write_index] = A[read_index];
            write_index--;
        }
        read_index--;
    }

    while(write_index >= 0)
    {
        A[write_index] = 0;
        write_index--;
    }
}
```

a) Perform the complete dry run on the given algorithm and show array content on each iteration. [6 marks]
Assume contents of array are {1, 10, 20, 0, 59, 63, 0, 88, 0}.

Iteration	Array
	First loop:
1	read = 7, write = 8
2	{ 1, 10, 20, 0, 59, 63, 0, 88, 88 } read = 6, write = 7
3	" " read = 5, write = 7
4	{ 1, 10, 20, 0, 59, 63, 0, 63, 88 } read = 4, write = 6
5	{ 1, 10, 20, 0, 59, 63, 59, 63, 88 } read = 3, write = 5
6	" read = 2, write = 5
7	{ 1, 10, 20, 0, 59, 20, 59, 63, 88 } read = 1, write = 4
8	{ 1, 10, 20, 0, 10, 20, 59, 63, 88 } read = 0, write = 3
9	{ 1, 10, 20, 1, 10, 20, 59, 63, 88 } read = -1, write = 2
	Second loop:
1	{ 1, 10, 0, 1, 10, 20, 59, 63, 88 } read = -1, write = 1
2	{ 1, 0, 0, 1, 10, 20, 59, 63, 88 } read = -1, write =
3	{ 0, 0, 0, 1, 10, 20, 59, 63, 88 } read = -1, write

6

b) What is the purpose of the given algorithm? [2 Marks]

Purpose of given algorithm is to find all zero in the given array and move zero to the start of array.

c) What will be the final output of foo()? [1 Mark]

{ 0, 0, 0, 1, 10, 20, 57, 63, 88 }

d) What is the complexity of given code in terms of Big-Oh? [2 Marks]

Complexity of given Code O(n)

e) What will be the best-case scenario for the given algorithm? [2 Marks]

All zero present at the end of array or There is no non-zero term

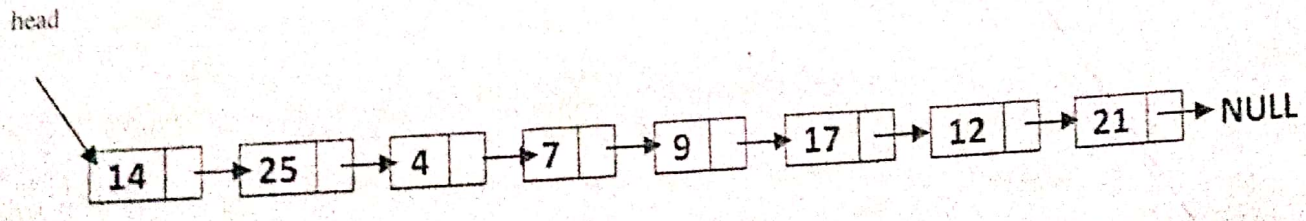
Question 3 [20 Marks]

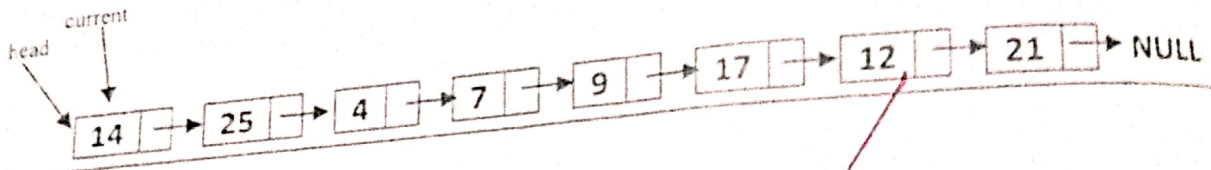
1. Consider the following code snippet:

```
void list::Game1()
{
        node* headref = head;
        genrated = NULL;
        node* current = headref;
        while (current != NULL) {
                node* next = current->next;
                Game2(current);
                current = next;
        }
        head = genrated;
}

void list::Game2(node* newnode)
{
        if (genrated == NULL || genrated->data >= newnode->data) {
                newnode->next = genrated;
                genrated = newnode;
        }
        else {
                node* current = genrated;
                while (current->next != NULL
                        && current->next->data < newnode->data) {
                        current = current->next;
                }
                newnode->next = current->next;
                current->next = newnode;
        }
}
```

a) Given a linked list (given below), perform a complete dry run of the algorithm and at each iteration, display the structure of linked list. [7 Marks]



Iteration	
	generated=NULL 

b) What is the wors complexity of the above code in terms of Big-Oh? Justify your answer! [1 Marks]

c) What is the best-case time complexity of the above algorithm? Justify your answer! [2 Marks]

d) What is the purpose of the given algorithm? and write down the name of algorithm. [2 Marks]

2. Given the following code of selection sort.

```
void Func()
{
        node* ptr_1, * ptr_2, * min;
        ptr_1 = head;
        while (ptr_1->next != NULL)
        {
                ptr_2 = ptr_1;  <-- Statement is missing here
                min = ptr_1;    <-- Statement is missing here
                while (ptr_2 != NULL)
                {
                        if (min->data > ptr_2->data)
                                min = ptr_2;
                        ptr_2 = ptr_2->next;
                }
                int tmp = ptr_1->data;
                ptr_1->data = min->data;
                min->data = tmp;
                ptr_1 = ptr_1->next;  <-- Statement is missing here
        }
}
```

3

a) Add the missing lines of code to the above-mentioned function. [3 Marks]

b) What is the complexity of the above code in terms of Big-Oh? [1 Marks]

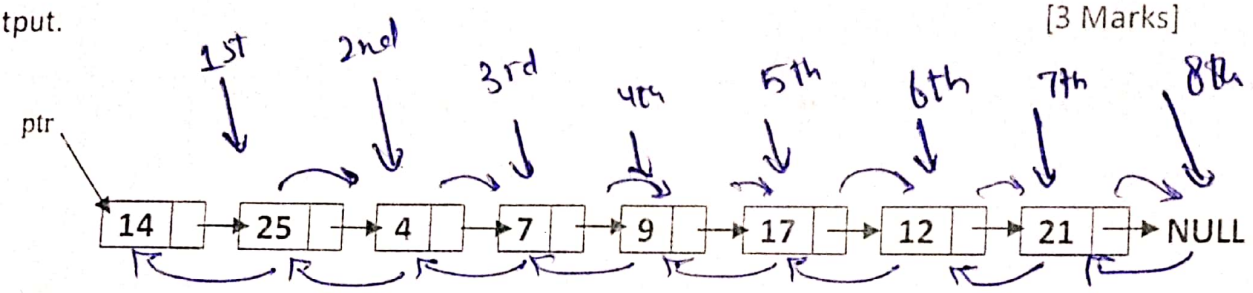
$O(n^2)$

1

3. Consider the following function.

```
void fun(node* ptr)
{
        if (ptr == NULL)
                return;
        fun(ptr->next);
        if(ptr->data%2==0)
                cout << ptr->data << " ";
}
```

a) Given a linked list (given below), perform the complete dry run of the algorithm and display the output. [3 Marks]



Output:

12 4 14

3

b) What is the complexity of the above code in terms of Big-Oh? [1 Marks]