University Roll No. - T91/ECE/204058

Subject - COMPUTER SCIENCE & ENGINEER

Semester - 2nd

Paper Code - CS204

Date-of-Examination - 16.08.2021

Group - A [5 Questions]

(1.) a)

	LINEAR Data Structure	NON-LINEAR Data Structure
1.	In linear Data Structure, data elements are arranged squentially and in a linear fashion where every elements are attached to ~~the~~ its previous and next adjacent data element.	Here, data elements are not stored / arranged sequentially or linearly. It is more efficient in utilizing computer memory in comparison to linear DS.
2.	Traversal is easy and hence, it is easy to implement.	Here, ~~we can't~~ traversal to all elements is not possible in a single rein. Hence, it is quite difficult to implement
3.	Examples: Linked list, Arrays, Stack, queue.	Examples: graphs, trees, Binary Search Trees.

b)

	PRE - INCREMENT	POST - DECREMENT
1.	Pre - increment, increments the value of a variable before using it in an expression.	Post - decement, decrements the value of variable after executing the expression completely.
2.	Syntax : a = ++x ;	Syntax : a = x-- ;

Example : C Program to demonstrate pre and post decrement :-

```
# include <stdio.h>
int main() {
    int x = 10, a = 0, b = 0;
    a = ++x;        // pree increment
    b = x--;        // post decrement

    printf("%d", a);    // O/P = 11 (x is incremented and then assigned to a)
    printf("%d", b);    // O/P = 11 (x is first assigned to b then decremented)
    printf("%d", x);    // O/P = 10

    return 0;
}
```

c)

	RETURN	EXIT
1.	Keyword "return" causes the current function to return the control to the calling fxn.	Exit function → terminates the execution of program and returns the control to the operating system.
2.	This requires the header <stdio.h>	This requires the header <stdlib.h> (The library to use exit() function)

(d) → Prefix expression is an expression in which the operator appears in the expression before the operands.

→ Eg: If, Infix expression = A+B
then prefix expression = +AB [prefix notation]

→ This is of the form : (operator operand1 operand2)

More eg : Infix : ((A+B) * (C-D))/E
prefix : /*+AB-CDE

(e) Array is a linear data structure that can store fixed size sequential collection of elements (contiguous memory locations) of similar data types and its data elements can be accessed randomly using indices (0-based) of an array.

It can store collection of primitive data types such as int, float, char, double etc...

Array declaration syntax: int a[10]; // an array with 10 elements of integer datatype.

Group B (6 questions)

(3) // C program to copy file contents from a file to another file :-

```
#include <stdio.h>
#include <stdlib.h>
int main() {
FILE *fs, *ft;
char ch;
fs = fopen("io.txt", "r");   // let's say we copy file "io.txt" to "out.txt"
```

```
if (fs == NULL) {
        printf (" ERROR! Cannot open source file ");
        exit(1);
}
ft = fopen ("out.txt", "w");    // let's say the contents are copied to "out.txt"
if (ft == NULL) {
    printf (" ERROR! Cannot open target file");
    fclose (fs);
    exit(1);
}
while(1) {
  ch = fgets( fs);
  if (ch == EOF)
        break;
  else
     fputc (ch, ft);
}
fclose (fs);
fclose (ft);
return 0;
}
```

(6.) // C program to print prime numbers within 1 to 100 (using sieve)

```
#include <stdio.h>

void SieveOfEratosthenes (int start, int end) {
int prime[end];
int i=0, j=0;
for(i=2; i<=end; i++)
    prime[i] = 0;
for (i=2; i<=end; i++) {
    if (prime[i] == 0) {
        for (j= i*i; j<=end; ++j)
            prime[j] = 1;
    }
}
printf ("The prime numbers within the range [%d, %d] is ", start, end);
for(i=2; i<=end; i++){
    if (prime[i]==0 && i>=start && i<=end)
        printf ("%d", i);
}
}
int main() {

int start=0, end=0;
printf("enter the range: ");
scanf ("%d%d", &start, &end);

start -= (start<0) * 2 * start;   // if start is negative, make it positive
end -= (end<0) * 2 * end;

if (start > end) {                // swap if start > end.
    start ^= end;
    end ^= start;
    start ^= end; }
```

```
Sieve(start, end);
return 0;
}
```

Time complexity : (@ worst case) : $O(n * \log(\log n))$.

OR

(7.) // C Program to compute sum of first 10 even numbers.

```
# include <stdio.h>
int evenSum (int n) {
    int i=0, j=2, sum=0;
    for (i=1; i<=n; i++) {
        sum += j;
        j += 2;        // to get next even no.
    }
    return sum;
}
int main () {
    int n = 10;
    printf ("The sum of first 10 even numbers is %d", evenSum(n));
    return 0;
}
```

OR using Formula.

```
# include <stdio.h>
int evenSumFormula (int n) {
    return ((n+1)*n);
}
int main () {
    int n = 10;
    printf ("The sum of first 10 even numbers is %d", evenSumFormula(n));
    return 0;
}
```

(8.) //C Program to find out whether a year is a leap year or not.

```
# include <stdio.h>

int checkLeap (int year) {
    if (year % 400 == 0)
        return 1;
    if (year % 100 == 0)
        return 0;
    if (year % 4 == 0)
        return 1;
return 0;
}

int main() {
    int year = 0;
    printf ("Enter the year : ");
    scanf ("%d", &year);
    prin
    if (checkLeap (year)
        printf (" YES! It's a leap year.");
    else
        printf (" OPPS! This year is not a leap year.");
    return 0;
}
```

I/P : Enter the year : 2021

O/P : OPPS! This year is not a leap year.

I/P : Enter the year : 2000

O/P : YES! It's a leap year.

(9.) // C program to find out the GCD of 2 integers.

```
# include <stdio.h>
int GCD (int x, int y) {
    int min = 0;
    if ( x > y)          // checking minimum
        min = y;
    else
        min = x;
    for( int i = min ; i >= 1 ; --i) {
        if ( x % i == 0 && y % i == 0)
            return i;
    }
    return 0;
}
int main() {
    int x = 0, y = 0;
    printf (" Enter two numbers: ");
    scanf (" %d %d ", &x, &y);
    if (GCD(x,y))
        printf (" HCF found = %d ", GCD(x,y));
    else
        printf (" Invalid input ");
    return 0;
}
```

I/P : Enter two numbers : 4 8

O/P : HCF found = 4

(10) // C Program to print the Fibonacci series.

```
# include <stdio.h>

void printFib (int n) {
    int f1 = 0 , f2 = 1 , i = 0;
    if (n < 0)
        return ;
    printf ("%2d ", f1);
    for ( i = 1 ; i < n ; i++) {
        printf ("%2d ", f2);
        int next = f1 + f2;
        f1 = f2;
        f2 = next;
    }
}
int main () {
    int n = 0;
    printf (" Enter a number : ");
    scanf ("%d", &n);
    printf (" Fibonacci Series is : ");
    void
    printFib (n);
    return 0;
}
```

I/P: Enter a number : 6
O/P: Fibonacci Series is : 0 1 1 2 3 5

Group C (3 ques.)

(14.)

```
# include <stdio.h>
# include <ctype.h>
int countSpaces(char *s) {          // counts the number of spaces in str
    int c = 0;
    while (*s != '\0') {
        if (*s == ' ')
            c++;
        s++;
    }
    return c;
}

int
void mk_initial (char* s) {
    int c = countSpace(s);          // counts total spaces.
    while (*s) {
        printf ("%c. ", toupper(*s));
        while (*s != 32)            // skip character until ' ' is found
            s++;
        c--;                        // decrement count.
        if (c == 0)
            break;
        s++;                        // skip spaces
    }
    while (*s) {
        printf ("%c", *s);
        s++;
    }
}
```

```
int main() {
    char str[50];
    printf("Enter a string: ");
    scanf("%[^\n]s", str);
    mk_initial(str);
    return 0;
}
```

(15.) a) A structure is is a user defined datatype in C language. It creates a datatype that can be used to group items of different data types.

"Struct" keyword is used to create a structure in C. When a structure is declared, no memory is allocated for it. Memory is allocated when variables are created.

	Array of structure	Structure of array
1.	Array of structure is an array whose data elements are of the new data type of structure	A structure that contains an array as its member, is called as an structure of array or array within a structure.
2.	Access: Can be accessed by indexing similar to how we access an array.	Access: It is accessed using the dot (.) operator just as we access other elements of a structure.

3.	Syntax:
	`struct class {`
	`int a, b, c;`
	`} students[5];`
		Syntax:
		`struct class {`
		`int arr[10];`
		`} a, b;`

(5) // C program to find maximum and minimum element in an array.

```
#include <stdio.h>
int main() {
    int n, max=0, min=0, temp=0;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    printf("Enter n numbers: ");
    for(int i=0; i<n; i++) {
        int temp=0;
        scanf("%d", &temp);
        if(i==0) {
            max = temp;
            min = temp;
        }
        else {
            if(temp > max)
                max = temp;
            else if(temp
            if( temp< min)
                min = temp;
        }
    }
    printf("Largest element: %d", max);
    printf("Smallest element: %d", min);
    return 0;
}
```

(12.) Advantages of a linked list over arrays are :-

i) The size of array is fixed but ~~the size of~~ linked list is of dynamic size. So, size is not an issue in linkedlist.

ii) Insertion and deletion of an element ~~at~~ is easier in linkedlist without reallocation or reorganisation of entire structure as in case of array.

iii) Time complexity for insertion at start is $O(1)$ & Time complexity for insertion at end is $O(n)$ in case of a linkedlist.

Insertion at beginning of Single linked list :-

2 steps :-

(i) Make the next pointer at the node towards the first node of the linkedlist.

(ii) Make the start pointer point towards this new node.

[~~If the~~ check : if the linkedlist is empty, make the start pointer point to the new node made.

Implementation function

```
void insertAtHead (node *p, int val) {
    struct node * temp = malloc (sizeof (struct node));
    if (start == NULL) {  // empty
        start = p;
        printf ("\n node inserted");
    }
    else {
        temp = start;
        start = p;
        p -> next = temp;  // new node points to the previous
                           1st node.
    }
}
```