1. How to increment a pointer variable? Write a sample program for it.

Pointers variables are also known as address data types because they are used to store the address of another variable. The address is the memory location that is assigned to the variable. It doesn't store any value. Hence, there are only a few operations that are allowed to perform on Pointers in C language. The operations are slightly different from the ones that we generally use for mathematical calculations.

Increment a pointer

* It is a condition that also comes under addition.
* When a pointer is incremented, it actually increments by the number equal to the size of the data type for which it is a pointer.

Example:

If an integer pointer that stores address 1000 is incremented, then it will increment by 2 (size of an int) and the new address it will point to 1002. While if a float type pointer is incremented then it will increment by 4 (size of a float) and the new address will be 1004.

Program:

```
#include <stdio.h>
int main ()
{
  int N=4;
  int *ptr1, *ptr2;
  ptr1= &N;
  ptr2= &N;
 printf ("Pointer ptr1' before increment:");
 printf ("%p \n", ptr1);
 ptr1++; // Incrementing pointer ptr1;
 printf ("Pointer ptr1 after increment: ");
 printf ("%p \n", ptr1);
 return 0;
}
```

Output:

```
Pointer ptr1 before increment: 0x7ffefa9c5704
Pointer ptr1 after increment: 0x7ffefa9c5708
```

2. List the various string handling functions. Explain any 2 with example.

C programming language provides a set of predefined functions called string handling functions to work with string values. The string handling functions are defined in a header file called string.h. Whenever we want to use any string handling function we must include the header file called string.h.

Function	Syntax	Description
strcpy()	strcpy (string1, string2)	Copies string2 value into string 1.
strncpy()	strncpy (string1, string2, 5)	Copies first 5 characters string 2 into string 1.
strlen()	strlen(string1)	returns total number of characters in string 1.
strcat()	strcat(string1, string2)	Appends string 2 to string 1.
strncat()	strncat(string1, string2, 4)	Appends first 4 characters of string 2 to string 1.

strcmp()	strcmp(string1, string2)	Returns 0 if both strings are same. less than 0 if string1 < string 2; greater than 0 if string1 > string 2.
strcmpi()	strcmpi (string1, string2)	Compares 2 strings, by ignoring case.
stricmp()	stricmp (string1, string2)	Compares 2 strings by ignoring case.
strlwr()	strlwr (string1)	Convert all the characters to ~~st~~ lower case.
strupr()	strupr(string1)	Convert all the characters to upper case.
strdup()	string1 = strdup(string2)	Duplicated value of string 2 is assigned to string 1.
strchr()	strchr (string1, 'b')	Returns a pointer to the first occurrence of 'b' in string 1.
strstr()	strstr (string1, string2)	Returns a pointer to the first occurrence of string2 in string1.
strset()	strset(string1, 'B')	Sets all the characters of string 1 to given character 'B'.
strrev()	strrev (string1)	It reverses the value of string1.

Example 1: strcat()

```
#include <stdio.h>
#include <string.h>
int main()
{
 char source[]=" Hello";
 char target[]= "World";
 Strcat(~~source~~ target, ~~target~~ source);
 printf(" Target string after strcat() = %S", target);
}
```

output: Target string after strcat() = Hello World.

Example 2: Strlen()

```
#include <stdio.h>
#include <string.h>
int main()
{
 char a[20];
 Printf("Enter String: ");
 Scanf("%S", &a);
 printf("Length of string = %d \n", strlen(a));
}
```

output:
```
Enter string: Function
Length of string= 7
```

3. Define structure. How to access its members? Give an example for it.

→ Structure is a user defined data type available in C that allows to combine different data types of data items.

→ To create a structure, struct keyword must be used.

→ A struct in C language is a composite data type declaration that defines a physically grouped list of variables under one name in a block of memory, allowing the different variables to be accessed via a single pointer or by the struct declared name which returns the same address.

→ Structures are used to represent a record.

Syntax:

```
struct [structure tag]
{
 member definition;
 member definition;
 ...
} [one or more structure variables];
```

→ The structure tag is optional and each member definition is a normal valid variable definition, such as int a, float avg, char str.

→ At the end of structure's definition. It is optional to specify one or more structure variables.

Accessing Structure Members

→ Individual structure members can be used like other variables of the same type.

→ Structure members are accessed using the structure member operator (.), also called the dot operator, between the structure name and the member name.

The syntax for accessing the member of the structure is :

```
structurevariable.member-name;
```

Ex:
```
struct coordinate
{
  int x;
  int y;
};
```

Thus to have the structure named first refer to a sreen location that has x coordinates x=50, y=100, it can be written as,

```
first.x = 50;
first.y = 100;
```

Program:

```
#include <stdio.h>
#include <string.h>
struct Books
{
 char title[50]; char author [50];
 Char subject [100]; int book_id;
 };
int main ()
{
  struct Books Book1;
  struct Books Book2;
  strcpy(Book1.title, "C Programming");
  strcpy (Book1.author, "Nuha Ali");
  strcpy(Book1.subject, "C programming Tutorial");
  Book1. book_id = 6495407;
  Strcpy (Book2. title, "Telecom Billing");
  strcpy (Book2. author, "Zara Ali");
  strpy (Book2. subject, "Telecom Billing Tutorial");
  Book2. book_id = 6495700;
 printf(" Book 1 title : %s \n", Book1. title);
 printf ("Book 1author : %s \n", Book1. author);
 printf ("Book 1 subject : %s \n", Book1. subject);
 printf ("Book1 book_id : %d \n", Book1. book_id);
printf("Book 2 title : %s \n", Book2.title);
printf ("Book 2 author : %s \n", Book2. author);
printf ("Book 2 subject: %s \n", Book 2. subject);
printf("Book 2 book_id: %d \n", Book2. book_id);
return 0;
}
```

Output:

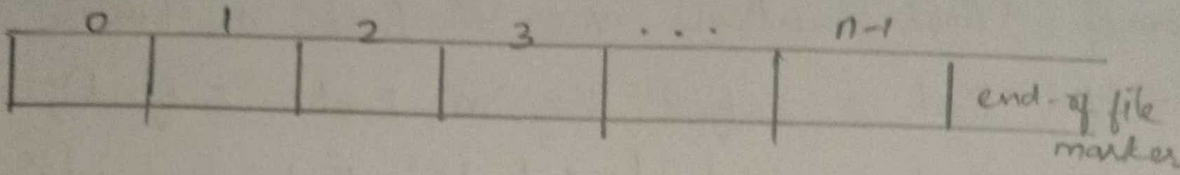
```
Book 1 title: C programming
Book 1 author: D.Nuha Ali
Book 1 subject: C programming Tutorial
Book 1 book_id: 6495407

Book 2 title: Telecom Billing
Book 2 author: Zara Ali
Book 2 subject: Telecom Billing Tutorial
Book 2 book_id: 6495700.
```

4. Write the concept of Sequential and Random Access Files.

Sequential Access Files

C views each file simply as a sequential stream of bytes. Each file ends either with an end-of-file marker or at a specific byte number recorded in a system-maintained, administrative data structure.

When a file is opened, a stream is associated with it.



Streams provide communication channels between files and programs. Three files and their associated streams are automatically opened when program execution begins:

* the standard input to read data from the keyboard.
* the standard output to print data on the screen.
* the standard error.

Creating a Sequential- Access File

This program assumes the user enters the records in account-number order. The records would be sorted and written to the file.

Program:

```
# include <stdio.h>
 int main (void)
 { unsigned int account ;
   char name [30] ;
    double balance ;
     FILE  * cfPtr;
   if ((cfPtr = fopen ("client. dat", "w")) == NULL)
   { puts ('File could not be opened');
   }
else
{
puts ("Enter the account, name and balance ");
puts ("Enter EOF to end input.");
printf( "%s", "?");
scanf (" %d %29s %lf", &account, name, &balance);
while (! feof (stdlin))
{
  fprintf(cfptr, "%d %s % .2f \n", account, name,
                                    balance);
  printf("%s", "?");
  scanf("%d %29s %lf"}; &account, name, &balance);
}
fclose (cfPtr);
}
}
```

Output:

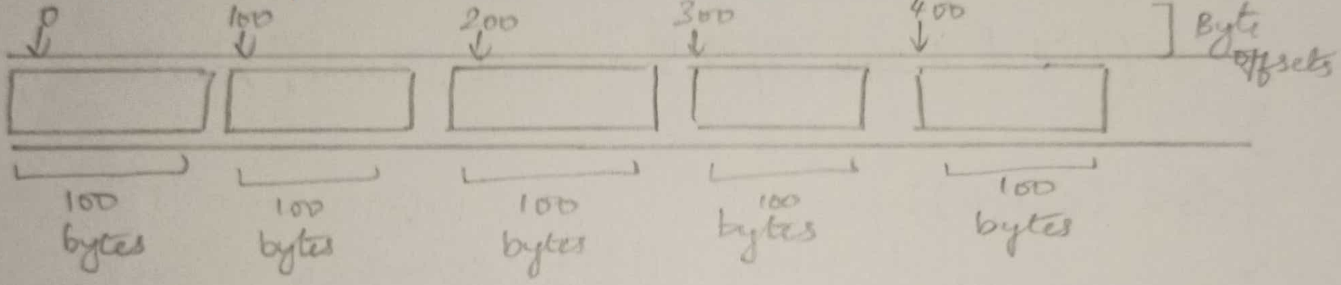
```
Enter the account, name and balance.
Enter EOF to end input.

  100  Jones  24.98
  200  Doe    345.67
  300  White   0.00
  400  Stone  -42.16
  500  Rich   224.62
```

Random - Access File.

Individual records of a random - access file are normally fixed in length and may be accessed directly without ~~seat~~ searching through other records.

This makes random - access files appropriate for:

* banking system
* point-of- scale system.
* other kinds of transaction - processing systems that require rapid access to specific data.



Fixed - length records enable data to be inserted in a random - access file without destroying other data in the file. Data stored previously can also be updated or deleted without rewriting the entire file.

Creating a Random Access File.

The following program open a random - access file, define a record format using a struct, write data to the disk and close the file.

Program:

```
# include <stdio.h>
Struct clientdata
{
```

```
unsigned int acctnum;
char lastName [15];
char firstName [10];
double balance;
};
int main (void)
{unsigned int i;
 struct clientdata blankclient = {0," "," ", 0.0};
 FILE *cfPtr;
 if ((cfptr = fopen ('credit.dat","Wb")) == NULL)
 {puts ("File could not be opened");
 }
 else
 {
  for (i=1; i<=100 ; ++i)
  { fwrite (&blankclient, sizeof (struct clientData),
                                           1, cfPtr);
  }
  fclose (cfPtr);
  }
}
```