Unit - 4

Sorting :- sorting a technique of arranging the array element in a specified order.
i.e. either ascending or descending order.
for example :-

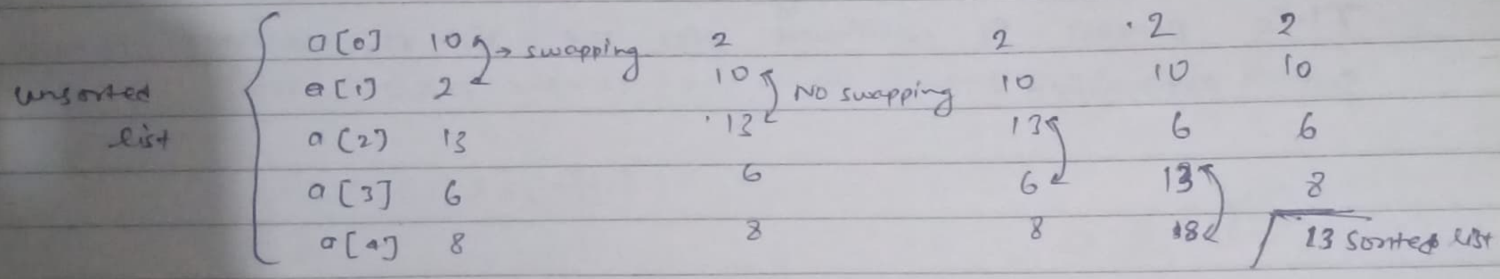
Array [ ] = { 3, 13, 2, 7, 26, 6 } unsorted array

= { 2, 3, 6, 7, 13, 26 } sorted array

there are several sorting Algorithm. Available some of them are, bubble sort, selection sort, Insertion sort, merge sort, shell sort etc.

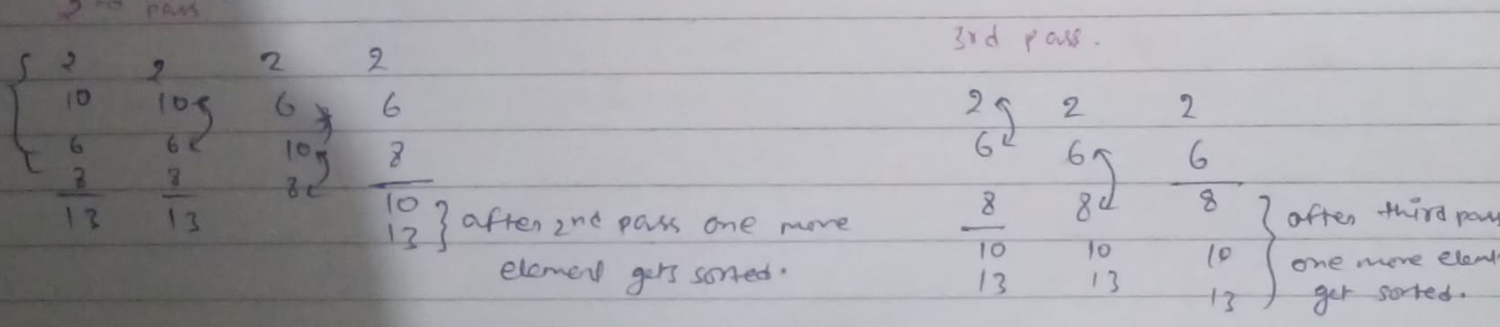
Bubble Sort => ~~No. of elements~~ ~~No. of~~

first pass



2nd pass

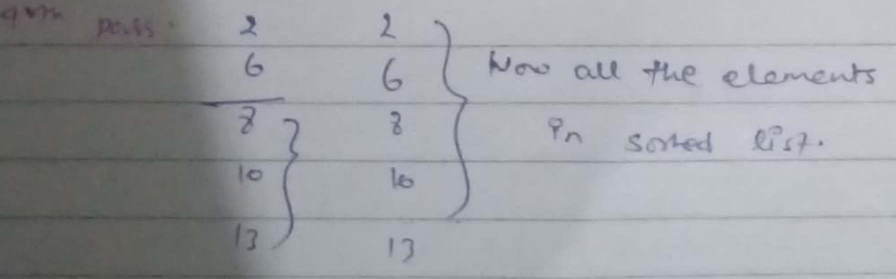
3rd pass.



after 2nd pass one more element gets sorted.

after third pass one more element get sorted.

4th pass.



Now all the elements are in sorted list.

No of element	No of maximum passes	
Eg. 5	4	
Eg n	n-1	
	Comparison	
first pass	4	often Every passes
Second pass	3	the comparison is
third pass	2	Reduce by 1.
fourth pass	1	

In bubble sorting consecutive adjacent pair element are compared with each other, if the element at the lower index is greater than the element at higher index, the two elements are interchanged. So, that the smaller element is placed before the bigger one.
This process is continued till the list of unsorted elements gets exhausted.

~~AA~~

Ques. Write a program to sort n number using bubble sort.

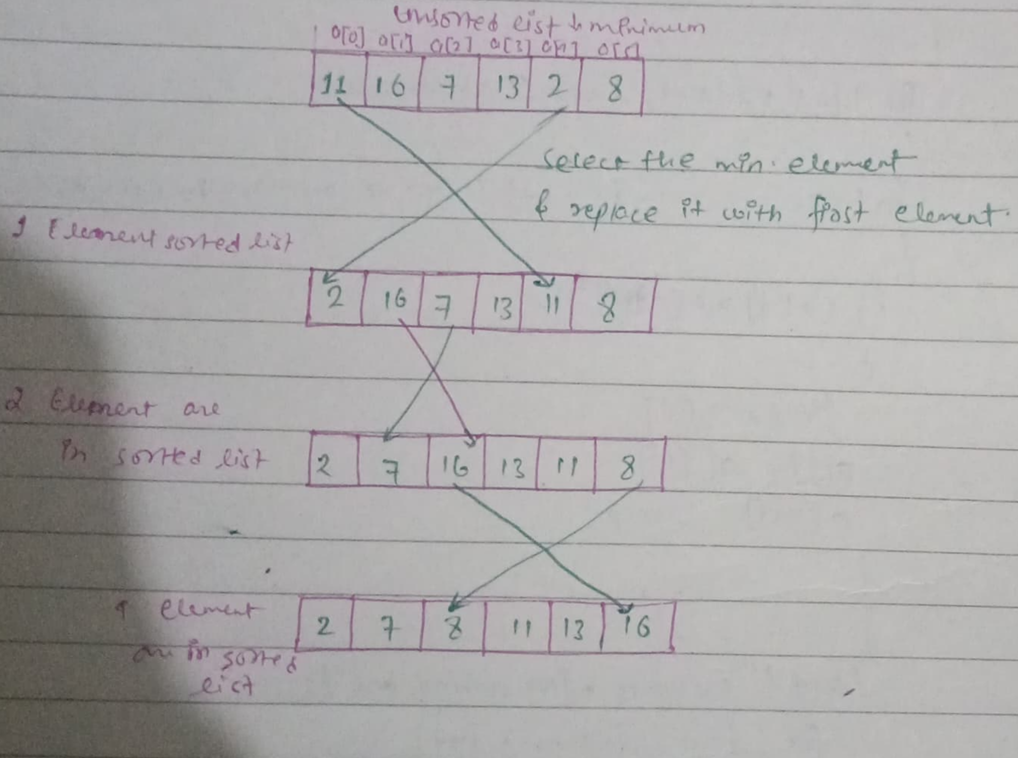
```
# include <stdio.h>
# include <conio.h>
  void main()
{ int a[100], n, i, j, temp;
 printf(" How many elements maximum 100");
 scanf(" %d", &n);
 printf(" enter elements");
  for (i=0; i<=n-1; i++)
  {
   scanf(" %d", &a[i]);
  }
   for (i=0; i<n-1; i++) // No. of passes
   {
    for(j=0; j<n-1-i; j++) // No. of comparison in each pass
    {
     if (a[j] > a[j+1])
     {
       temp = a[j]
      a[j]= a[j+1]
      a[j+1] = temp;
     }
    }
    Printf (" elements after sorting are");
     for (i=0; i<=n-1; i++)
     { printf ("%d", a[j]);
      }
        getch ();
      }
```

9286 o

Unsorted list :— The selection works on the principle of sorting by selection. The given unsorted list is initialy divided into ~~the~~ two list. the sorted list containing no element and unsorted list containing all elements.

Selection sort select the minimum element from the unsorted list and exchange it with the first element in the unsorted list. So the size of sorted list is increase by 1 and size of unsorted list decrease by 1.

For Ex :—



imp
★★

Que I - WAP to sort n numbers using selection sort.

Ans

```
        # include <stdio.h>
        # include <conio.h>
         void main( )
      { int a[100], n, i, j , temp, min;
       Printf("How many numbers maximum 100");
       Scanf ("%d", &n);
       Printf ("Enter numbers");
        for ( i=0; i<=n-1; i++)
        {
          Scanf ("%d", &a[i]);
        }
            for( i=0; i<n-1; i++)
            {  min = i;
        for ( J= i+1; j<=n-1; j++)
            {
              if (a[min] > a[j])
                 min = j;
            }
             if (min != i)
            {
               temp = a[i];
                a[i] = a[min];
                a[min] = temp;
             }}}
                Printf ("Elements after sorting are");
                for ( i=0; i<=n-1; i++)
                 {
                   Printf("%d", a[i]);
                 } getch();
                   }
```

Searching ⇒ Seaching is the process of Finding some particular element is the list, if element is present in the list then search is successful and ~~[illegible]~~ return the location of that particular elements other wise search is unsuccessful.

-: There are two seaching algorithm :-

(1) linear or sequential search.

(2) Binary Search.

(1) Linear or Sequential Search :- in this we simply traverse the list completely and match each element of the list with the element whose location is to be found. If match is found then return the location of that particular element otherwise algorithm returns null.

e.g.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
50	3	13	225	55	200	150	55	10

Sea- $\boxed{55}$

Imp PUT

Que- WAP to search a element in Array using linear search.

Ans ⇒:

```
# Include <stdio.h>
# Include <conio.h>
void main()
{ int a[100], n, i, key, found = 0;
Printf("How many elements, maximum 100");
Scanf("%d", &n);
 Printf("Enter numbers");
 for (i=0; i<=n-1; i++)
  {
   Scanf("%d", &a[i]);
  }
  Printf("Enter the Elements that you want to search");
 Scanf("%d", &key);
   for (i=0; i<=n-1; i++)
    {
      if (a[i]== key)
     { found = 1
      Printf("%d found", key);
     Printf("location = %d", i+1);
    }
    }
      if (found ==0)
      Printf("%d not found", key);
     getch();
       }
```

(2) Binary Search ⇒ → Binary search works on the principle of divide & conquer.

→ For this algorithm data should be in sorted form (order).

→ this algorithm much faster as compared linear search algorithm.

E.g.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
2	10	23	36	54	78	98	102

first = 0 } first 0

last = 7 } last = n-1

middle = (first + last) / 2

= (0 + 7) 2

= 3

36 — (i) middle

98 — Greater than middle

10 — less than middle.

Qu. WAP to search a element in array using Binary search.

```
# include <stdio.h>
# include <conio.h>
void main ( )
{ int a[100], n, i, key, first, last, middle;
  Printf ("How many elements, maximum 100");
  Scanf (" %d", &n);
  Printf ("Enter elements");
  for (i=0; i<=n-1; i++)
   {
    ~~Scanf ("Enter the elements for search");~~
    Scanf(" %d", & a [i]);
   }
   Printf("Enter the element for search");
   Scanf(" %d", & key);
     first = 0;
     last = n-1;
    middle = (first + last)/2
    while (first <= last)
    {
      if (a [middle] < key)
      { first = middle +1;
       }
       else if (a [middle] == key)
       {
        Printf (" %d found", key);
        Printf (" Location = %d", middle + 1);
          Break;
       }
        else
{ last = middle - 1;
  }
   middle = (first + last)/2;
  }
   if (first > last)
  Printf("%d not found", key);
   getch();
    }
```

Function :—

→ A function is the set of statement that ~~a~~ Perform a specific task and return a value as a result,

→ functions are subprograms which are used to perform a specific task, as they cannot run independently so, they are called by the main() function. or some other function.

- Advantages of function =
- Using function (modules) a large problem can be broken into smaller one so, that the problem can be solved easily.
- Reuseability of ~~a~~ code.
- Easy in debugging and testing.
- In 'C' language function are divided in two parts.

(1) Library function ~~or~~ ~~In~~ Built-in-functions ⇒ These are pre defined functions in c language. ~~Ex~~

——→

Ex- Scanf( ), printf( ), getch( ) etc.

2. User define function ⇒ The functions defined by the user according to requirement is known as user define function.

function Prototype or function declaration ⇒ it is neccessary to declare a function Before using it. it informs the compiler that the function would be use in the program at letter stage.

Syntax:-

```
return-type function-name (argument list);
```

E.g. (i) void message ( );
(ii) void message (int);
(iii) int message (void).
(iv) int message (int);

function Definition ⇒ function definition inform what a function does How it is being done.

Syntax:-

```
return-type function-name (Formal parameter optional)
{
    Body of function;
}
```

E.g.
```
# include <stdio.h>
# include <conio.h>
void main()   // calling function.
{ void message (); // function prototype
    message (); // called function
Printf (" I am in main () function");
    getch()
  }
    void message ()
  { printf ("It is easy to lean");
    }
```

Output
```
It is easy to leann
I am in main () function
```

Note:- 1) A function is called when function name is followed by semi collon.

2) A function is defined when function name is followed by pair of cleary braces.

Sum of two numbers
Using no argument no return value.

```
# include <stdio.h>
# include < conio.h>
  void main()
{ void sum ();
 sum (); // called function.
   getch();
   }

  void sum

{ int a,b,c;
 printf("Enter. two numbers");
 scanf ("%d%d ,&a,&b)
     c=a+b
 ~~pr~~ Printf(" Sum of two numbers);
```

Argument but no return value.

In this type of function, called function recieve dat from calling function

```
# Include <stdio.h>
# include <conio.h>
void main ()
{
 Void sum (int, int);
 int a,b ;
 Printf ("Enter two number");
 Scanf ("%d%d &a, &b)
      Sum (a,b)
      getch();
      }
```

```
=> void sum (int x , int y)
    {
     int z;
      z=x+y;
   Printf(" Sum of two no.= %d",z);
      }
```

No Argument but return value :-
In this type of function called Function does not receive any data from calling function but return result to calling function.

Qu- WAP to find the sum of two numbers using no argument but return value.

Syntax
Return Type
{variable = function name()}

Sol->
```
# include <stdio.h>
# include <conio.h>
void main ()
 { int sum();
   int s;
   S = Sum ();
Printf ("Sum of two numbers = %d", s);
  getch ();
  }
  int Sum ()
   { int a,b,c;
  Printf ("Enter two numbers");
 Scanf ("%d %d", &a, &b);
   c= a+b
  return c;
  }
```

* [illegible]

~~Paramete~~

Passing method — Sending value from calling function to called fuction is known parameter passing.

— There are two method —

1. Call by value. ↓

2. Call by Refrence. ↓

1. Call by value	2. Call by Refrence
• In call by value method copy value of actual paratmeter is passed to formal parameter.	• In call by Refrence method address of actual parameter is passed to formal parameter
• Any change in formal parameter will not reflect the value actual parameter.	• Any change in formal parameter will reflect the value of actual parameter.
• There is wastage of memory because actual and formal parameters have different memory location	• Memory can be save because actual and formal parameters ~~share~~ have the same memory location.
• This method is slow as compared to call by ~~value~~ Reference method	• This method is fast as compared to call by value.

~~Exam~~

Example of call by value —

• WAP to swap two numbers using call by value —

```
#include <stdio.h>
# include <conio.h>
void main ()
{ Void swap(int, int); // function prototype
   int a,b;
   Printf ("enter two numbers");
     Scanf ("%d %d", &a ,&b);
       Swap (a,b); // a&b are actual
                      parameter.
   Printf("value of actual parameter = %d %d", a,b);
   getch();
}
void swap (int x, int y) // x&y are
                              formal
                              parameter
{ int temp ;
   temp = x;
      x = y;
      y = temp;
 Printf("value of formal parameter = %d %d", x,y);
}
```

Ex of call by Reference :-

• WAP to swap two number using call by reference.

```
  # include <stdio.h>
   # include< conio.h>
   void main ( )
  { void swap (int* ,int *); // Function prototype
      int a,b;
     Printf ("Enter two numbers");
     Scanf ("%d %d",&a ,&b);
          Swap (&a ,&b); ~~#~~
    Print f(" value of actual parameter = %d %d", a ,b);
          getch( );
          }
          void swap ( int *x, int *y)
      {
            int temp;
            temp = *x ;
            *x = *y ;
            *y = temp;
       Printf (" value of formal parameter = %d %d", *x, *y);
          }
```

Storage class :- storage class indicates the place of storage, scope, lifetime and default initial value of a variable.

There are four storage class in 'C' language.

(1) Automatic storage class (auto)
(2) Register storage class (register)
(3) Static storage class (Static)
(4) External storage class (extern)

Storage class Specifier	Place of storage	Scope	Life Time	Default Initial value
1 auto	Primary Memory (RAM)	with in the function where it is declared or local scope	Exist from time of Entry in the function to the end of function	garbage
2. Register	Register of (CPU)	with in the function where it is declared or local scope	Exist from ~~the~~ time of Entry in the function to the End of function	garbage
3. Static	Primary memory (RAM)	with in function where it is declared or local scope.	Retain the value of variable between different function call	Zero
4. Extern	Primary Memory (RAM)	can be use in all the function of a program or global scope	Exist as long as program is in Execution	Zero

(1) Example of automatic storage class.

```
# include <stdio.h>
# include <conio.h>                    void f1()
void main()                            {int a=30
{  void f1();                           printf("%d",a);
     f1();                             }
  auto int a = 20;
  printf("%d", a);                       output 30,20.
     getch();
  }
```

② Example of Register storage class

```
# include<stdio.h>
# include<conio.h>
void main()
  { register int i
    for(i=1; i<=100; i++)
     {
       Printf ("Hello");
     }
     getch();
     }
```

③ Example of static storage class:

```
# —
# —
  void main()
  { static int x=5;
  Printf("%d", x--);
  if (x>0)
  main ();
  getch ();
  }
   output: 5,4 3 2 1.
```

⑤

Example of Auto, Register, static external
storage class:—

```
# include<stdio.h>
# include <conio.h>
~~void main()~~
   int x;
 void main()
{ auto int y;
  register int z;
  Static int m;
Printf("%d %d %d %d", x,y, z, m);
  getch();
  }        output:— 0, garbage, ", 0.
```

④ Example of External storage Class:

```
#  —
#  —
 int a;
 void main()
 { int b;
Printf("%d %d", a,b);
  getch();
  }
 output : zero, garbage.
```

~~No argu~~ (4) Argument with return value

In this type of function called function Receive data from calling function and also return result to calling function.

#

Que WAP to find sum of two number using argument with return value.

Sol)
```
# include <stdio.h>
# include <conio.h>
void main()
{ int sum (int, int);
  int a,b,s;
Printf ("Enter two numbers");
Scanf ("%d %d", &a, &b);
  S= Sum (a,b);
 Printf ("Sum of two numbers= %d", s);
  getch();
 }
      int sum (int x , int y)
        {
          return (x+y);
        }
```

Recursion:— Recursion is powerful programing technique. it is bisically used when a problem can be expressed in terms of similar problem of smaller size.

When a function called itself then this function is called recursive function and this process is known as recursion. ~~it is~~ ~~of two types~~ — it is of Two Types —

Direct Recursion	Indirect Recursion
Eg. A( )	A( )
{	{
=	=
A( );	B( );
}	}
	B( )
	{
	=
	A( );
	}

Que. WAP to find factorial of no. using Recursion.

```
Sol. #include <stdio.h>
     # include <conio.h>
     void main()
     { int factorial (int);
       int (n,f);
     Print f ("Enter a number");
     Scanf ("%d",&n);
       f= ~~[illegible]~~ factori al(n);
         f = factorial (n);
```

```
~~Print f ("factorial %d",f);~~
~~getch();~~
}
{ ~~int~~ ~~factorial (int~~
int n,f
int ~~f~~ factorial (int+no) )
{
  if (no. = = 1)
    return;
  else
  return ( no. * Factorial (no.-1));
  }
}
```

Que WAP to find the GCD (Greatest Common division of two number using recursion.

Eg:- {rem: remainder}

```
#—
#—
void main()
{ int GCD (int, int)
   int n1, n2 result;
Scanf ~~Printf~~ ("%d %d", &n1, &n2);
   result = ~~f~~ GCD (n1, n2);
   Printf ("GCD of two numbers = %d", result);
   getch();
   }
int GCD (int x, int y)
{ int rem.
   rem = x%y
   if rem = = 0
   return y;
else
   return (GCD (y, rem));
   }
```

Qu write to print fibonacci series up to n terms Using Recursion.

Sol^n,

```
#__
#__

void main ( )
{ int fibonacci (int);
   int n, f, i;
 Printf(" How many terms ? ");
 Scanf(" %d", &n);
  for (i=1 ; i<=n-1 ; i++)
   {
     f= fibonacci (i);
   Printf (" %d", f);
  }
   getch ();
    }

  int fib(int no.)
  {
    if (no. = = 1)
    return =0 ;
  else if (no= = 2)
    return 1;
 else
    return ( fib(no-1) + fib (no-2));
     }.
```

Structure:- Structure is collection of Heterogeneous Type of data elements i.e. collection of different data type group together.

Structure declaration:- We use keyword struct along with the user defined name which is followed by pair of { curly Braces that have different component/elements

Syntax:-

```
Struct User defined Name
  {
     data Type member1;
     data Type member 2;
     - - -
  };
```

☆ Qu WAP to create a structure for a Book details are title, author, Pages and price. Input the details of a Book and print.

```
# include <stdio.h>
# include <conio.h>
void main()
{
 Struct Book
   { char title [30];            } 
     Chare author [30];          } → Structure element
     int pages ;                 }
     float price ;               }
   } x;  → (variable element)
   Printf ("enter detail of a Book");
   Scanf (" %S %S %d %f ", x.title, x.author, & x.pages, &x.price);
   Printf (" details are ");
   Printf (" %S %S %d %f ", x.fittle, x.author, x.pages, x.price ");
        getch();
      }
```