Skill development

Assignment – 1

I. Aim->

Implement a menu driven Program in C++ using function for the following Array operations

a. Creating an Array of N Integer Elements
b. Display of Array Elements
c. Display sum of array elements.
d. Deleting an Element at a given valid Position(POS)

II. objective->

use iterative loop for finding sum all elemnts in array.
Perform the shifting the elemnts of array and delete the elemnt at given position.

III. Theory->

array in sequential data structure.
The elemnts are stored in consecutive memory location.
The size is defined at compile time.
To delete an element from an array ,shift all location of elemnt below that position one location upward and make last elemnt zero.

IV. Algorithm->

1) To accept an element in array.
    a. Declare an integer variable i.
    b. Declare an array of predefined size.
    c. Use for loop and accept the element in array.

2) To display element in array
    a. Declare an integer variable i.
    b. Use for loop for accessing all position in array .
    c. Print the elemnt in array.

3) To display sum of all element
    a. Declare an integer variable sum and equate it to zero.
    b. Use for loop for accessing all elements in array.
    c. Add each element to sum.
    d. After for loop display the sum.

4) To delete an element at given position.
    a. Accept the postion for deletion from user in pos variable.
    b. Use for loop from pos to size of array.
    c. Shift each element in array one position upward. After loop put zero in last positon in array.

V. <u>CODE-></u>

```
#include<iostream>
using namespace std;

void create(int arr[10],int);
void display(int arr[10],int);
void sum(int arr[10],int);
void deletearr(int arr[10],int);



void create(int arr[10] , int n)
{
cout<<"\nEnter the array elements \n";
for(int i=0;i<n;i++)
{
cin>>arr[i];
}
}

void display(int arr[10] , int n)
```

```
{
cout<<"\nThe array elements are\n";
for(int i=0;i<n;i++)
{
cout<<arr[i]<<"\t";
}
}
void sum(int arr[10],int n)
{
int sum=0;
for(int i=0;i<n;i++)
{
sum=sum+arr[i];
}
cout<<"\nSum of all array elements are \n";
cout<<sum;
}

void deletearr(int arr[10],int n)
{
int post;
cout<<"\nEnter the position to be deleted\n";
cin>>post;
if(post<=n)
{
for(int i=post-1;i<n;i++)
{
arr[i]=arr[i+1];
}
}
else
{
cout<<"\nInvalid \n";
}
cout<<"\nThe final array list is\n";
for(int i=0;i<n-1;i++)
{
cout<<arr[i]<<"\t";
}
}
int main()
{
int arr[10];
int n ;
cout<<"\nEnter the number of elements in array\n";
cin>>n;
create(arr,n);
display(arr,n);
sum(arr,n);
deletearr(arr,n);
cout<<endl<<endl;
return 0;
}
```

VI. <u>OUTPUT -></u>

```
Enter the number of elements in array
4

Enter the array elements
1
1
2
3

The array elements are
1	1	2	3
```

```
Sum of all array elements are
7

Enter the position to be deleted
2

The final array list is
1       2       3

Process returned 0 (0x0)   execution time : 12.159 s
```

VII. CONCLUSION->

The array is linear data structure in which values are stored in consecutive memory location.
To perform deletion in array is quite difficult .
We need to shift all the elements one position upward after deleting the element.

Skill development
Assignment – 02

1. **Aim:**
   Implement the following operations on string with pointers (without using library functions)
   a. Length
   b. Palindrome
   c. String Comparision

2. **Objective:**
   To understand the following Concepts : -
   i. Use of Pointers.
   ii. Call by reference.
   iii. Breaking the program into functions.

3. **Theory:**

i. **Pointers : -**
Pointers store the address of the variable. i.e. A pointer takes its value as the address in memory of another variable
  Syntax : -
    type *pointername;
&i – Gives the address of the i (Referencing operator &)
*j – Gives the value at address j (Indirection Operator *)
I — the value stored in that variable

ii. **Call by Reference : -**
Call by Reference is a method of calling function by using pointers. In this method we don't pass the value rather we pass the address of the variable as parameter to the function. So, all the operation are performed on that address and not on the variable.
  Syntax : -
    Function declaration
    return_type function_name(data_type *variable);

4. **Algorithm**
1. Start
2. Declare the function 'length' having return type integer and parameters as character pointer.

```
3. Declare the function 'paly' having return type void and parameters as two
   character pointer.
4. Declare the function 'comp'  having return type void and parameters as two
   character pointer.
5. Declare two char array 'str1' and 'str2' having size each of 10.
6. Display the menu.
      1) String Length
      2) Palindrome
      3) String compare

7. Accept choice of user from user and store it in 'ch'.
8. If the user enters '1'.
      i.   Accept string from the user and store it in 'str1'.
      ii.  Pass the str1 to length function.
      iii. Store the return value from length function in 'len'

         (Algorithm for 'length' Function with accepting local  parameter s1)
            1) Initialize variable integer 'i' as 0.
            2) Run a loop while value at 's' is not equal to NULL.
                  (within while loop)
                     a. Increment the value of i by 1.
                     b. Increment the value of s by 1
            3) Return i.
      iv.  Print 'len'.
9. If the user enters '2'.
      i.   Accept the string from user in str1.
      ii.  Pass to paly function.

         (Algorithm for 'paly' Function with parameter s1)
            1) Find the length of string accepted string and store in
               'lens1'.
            2) Initialize the i, count equal to 0 and j equal to lens1-1.
            3) Use while loop until s1[i]==s1[j].
                  (Within the while loop)
                  a. increment the count and  i by1 and decrement
                     the j by1
                  b. check If  i equals to lens1/2  then use break key
                     word.
            4) Check  If count equal to lens1/2
            5) If true Print given string is palindrome
            6) if false then print not palindrome

10. If the user enters '3'.
        i. Accept two strings from user and store them in 'str1' and 'str2'
           respectively.
       ii. Call the function comp with str1 and str2 as parameters.

    (Algorithm for 'comp' Function having accepting local parameters s1and s)

            1) Initialize integer variables 'add1', 'add2' and 'i' with
               value'0'.
            2) Run a while loop until s1[i] is not equal to NULL
                    (within while loop)
                        a. Add 'add1' and 's1[i]' and store it in add1
            3) Reinitialize 'i' as 0.
            4) Run a while loop until s2[i] is not equal to NULL.
                    (within while loop)
                        a. Add 'add2' and 's2[i]' and store it in add2.
            5) Check if  'add2' is greater than 'add1'
            6) If True, Print 's1 ' is greater.
            7) If False, check if 'add2' is less than 'add1'.
            8) If true, Print 's2' is greater.
            9) If False, Print 'They are equal'.
11. Continue the do while loop until the user enters 0 .
12. Stop.
```

5. Code : -

```
#include<iostream>
using namespace std;
int length(char *s);
void paly(char *s1);
void comp(char *,char *);
int main()
{
    int ch;
do{
      char str1[10],str2[10];
      int len;
cout<<"1:Length\n2:Palindrome\n3:String Comparision\n 0.exit\n enter your choice
:\n";
      cin>>ch;
      switch(ch)
            {
            case 1:
                  cout<<"\n Enter a string"<<endl;
                  cin>>str1;
                  len=length(str1);
                  cout<<"String Length = "<<len<<"\n"<<endl;
                  break;
            case 2:
                  cout<<"\n Enter a String -> ";
                  cin>>str1;
                  paly(str1);
                  break;
            case 3:
                  cout<<"\n Enter String 1 ->  ";
                  cin>>str1;
                  cout<<"\n Enter String 2 ->  ";
                  cin>>str2;
```

```
                        comp(str1,str2);
                        break;
                default:
                        cout<<"\nInvalid Option\n:";
                        break;
                }
        }
        while(ch!=0);
return 0;
}
int length(char *s)
        {
              int i=0;
                while(*s!='\0')
                {
                        i++;
                        s++;}
                return i;
        }

void paly(char *s1)
{
int lens1=length(s1);
int j=lens1-1, count=0,i=0;
while(s1[i]==s1[j])
{
i++;
j--;
count++;
if(i==(lens1/2))
break;
}
if(count==(lens1/2))
cout<<"\nPallindrome"<<endl;
else
```

```
cout<<"\nNot Pallindrome"<<endl;
cout<<endl;
}
void   comp(char *s1,char *s2)
{
      int add1=0,add2=0;
         int i =0;
         while(s1[i]!='\0')
         {
                  add1 =add1 + s1[i];
                  i++;
         }
         i=0;
         while(s2[i]!='\0')
         {
                  add2 =add2 + s2[i];
                  i++;
         }
         if(add2>add1)
                  cout<<s2<<" is greater"<<endl;
         else if(add2<add1)
                  cout<<s1<<" is greater"<<endl;
         else
                  cout<<"they are equal"<<endl;
                  cout<<endl;
}
```

6.OUTPUT->

```
1:Length
2:Palindrome
3:String Comparision
 0.exit
 enter your choice :
1
```

```
 Enter a string
shivam
String Length = 6

1:Length
2:Palindrome
3:String Comparision
 0.exit
 enter your choice :
2

 Enter a String -> abccba

Pallindrome

1:Length
2:Palindrome
3:String Comparision
 0.exit
 enter your choice :
3

 Enter String 1 ->  shivam

 Enter String 2 ->  ivam
shivam is greater

1:Length
2:Palindrome
3:String Comparision
 0.exit
 enter your choice :
3
```

```
 Enter String 1 ->  shivam

 Enter String 2 ->  pralhadrao
pralhadrao is greater

1:Length
2:Palindrome
3:String Comparision
 0.exit
 enter your choice :0
```

7.Conclusion->

a. from this assignment we came to know that how to pass the base address of character array and accept in pointer.
b. We came to know that how the implicate conversion from character to its ASS11 value takes place.
c. Hwo to perform the operations on pointer such as incrementation ,referencing , dereferencing .

Skill development
Assignment – 03

1. **Aim:**
   Implement the following operations on string with pointers (without using library functions)

   d. Copy
   e. Reverse
   f. Substring

2. **Objective:**
   To understand the following Concepts : -
   i. Use of Pointers.
   ii. Call by reference.
   iii. Breaking the program into functions.

3. **Theory:**

i. Pointers : -
Pointers store the address of the variable. i.e. A pointer takes its value as the address in memory of another variable
Syntax : -
```
type *pointername;
```
&i – Gives the address of the i (Referencing operator &)
*j – Gives the value at address j (Indirection Operator *)

ii. Call by Reference : -
Call by Reference is a method of calling function by using pointers. In this method we don't pass the value rather we pass the address of the variable as parameter to the function. So, all the operation are performed on that address and not on the variable.
Syntax : -
Function declaration
```
return_type function_name(data_type *variable);
```

4. Algorithm->

```
1) Start
2) Declare the function 'copy' having return type as integer and parameters as
   two character pointer.
3) Declare the function 'reverse' having return type as void and parameters as
   character pointer.
4) Declare the function 'substring' having return type as void and parameters as
   two character pointer.
5) Declare the function 'length ' having return type as integer and having
   parameters as character pointer.
6) Declare char array 'str1' and 'str2' each of size 10;
7) Declare variable 'len' and 'ch' as integer.
8) Display the menu.
      1. Copy
      2. Reverse
      3. Sub String
9) Accept choice of user from user and store it in 'ch'.
10)If the user enters '1'.
        i. Accept string from the user and store it in 'str1'.
        ii. Call the function 'copy' and pass 'str1' and 'str2' as parameter.

         (Algorithm for 'copy ' Function having s1 and s2 as parameter )
              1. Run a while loop until value at 's1' is not equal to
                 NULL.
                   (within while loop)
                 a. Copy the value of s1 to s2
                 b. Increment the value of s1 by 1.
                 c. Increment the value of s2 by 1.
        iii. Print str2.

11)If the user enters '2'.
        i. Accept string from the user and store it in 'str1'.
        ii. Call the function 'reverse' and pass 'str1' as parameters.

         (Algorithm for 'reverse' Function with parameter s)
              1. Declare a character variable 'temp'
              2. Declare variable integer 'len', 'j', 'i'.
              3. Assign value of 'len' as length(s)
              4. Assign value of 'j' as len – 1.
```

```
            5. Run a for loop from i equal to 0 to len /2
                    a. Assign value of  s[i] to temp.
                    b. Assign value of s[j] to s[i].
                    c. Assign value of temp as s[j].
                    d. Decrement the value of j by1.
                    e. Increment the value of  i by1.
            6. Assign value at s2 as '\0' (NULL).
    iii. Print 'str1' (Reversed String).

12)If the user enters '3'.
    i. Accept two strings from user and store them in 'str1' and 'str2'
       respectively.
    ii. Call the function 'substr' with str1 and str2 as parameters.

    (Algorithm for 'substr' Function with parameters 's1' and 's2')
        1. Initialize integer variables  'j' and 'i' with value '0'.
        2. Find the length of s1and s2 and assign to lens1and lens2
           respectively.
        3. Use a for loop from i=0 to lens1
                (Within for loop)
                    a. Check If s1[i] equals to s2[j]
                    b. If true then increment the count and  by 1
                    c. If false then execute next statment

        4. Check if count equals to lens2
                    a. If true then print substring found
                    b. If false then print substring not found

13)Stop.
```

5. Code : -

```
#include<iostream>
using namespace std;
int length(char *s);
void rev(char* );
void copy(char *s1,char *s2);
void sub(char *s1,char *s2);
int main()
{
        char str1[10],str2[10];
         int ch;

do
{
cout<<"\n1: Copy\n2:Reverse \n3:Substring  4.exit \nSelect Option:\n";
         cin>>ch;
         switch(ch)
                  {
                  case 1:
                           cout<<"\nEnter the string"<<endl;
                           cin>>str1;
                           copy(str1,str2);
                           cout<<"Copied String :- "<<str2<<"\n";
                           break;
                  case 2:
                           cout<<"\nEnter string."<<endl;
         cin>>str1;
         rev(str1);
         cout<<"\nReversed string is: "<<str1<<endl;
       break;
                  case 3:
                           cout<<"\nEnter a String :-   ";
                           cin>>str1;
```

```
                    cout<<"Enter the Substring:-   ";
                    cin>>str2;
                    sub(str1,str2);
                    break;
            default:
                    cout<<"\n Invalid Option\n:";
                    break;
            }
    }
    while(ch!=4);
return 0;
}
int length(char *s)
    {
            int i=0,len=0;
            while(*s!='\0')
            {
                    len++;
                    s++;
            }
            return len;
    }
void rev(char *a)
{
int len=length(a);
cout<<len;
int j=len-1;
for(int i=0;i<(len/2);i++)
{char temp;
temp=a[i];
a[i]=a[j];
a[j]=temp;
j--;
}
}
```

```
void copy(char *s1,char *s2)
        {
                while(*s1!='\0')
                {
                        *s2= *s1;
                        s1++;
                        s2++;
                }
                *s2='\0';
        }
void sub(char *a ,char *b)
{
int lens1=length(a);
int lens2=length(b);
int j=0,count=0;
for(int i=0;i<lens1;i++)
{
if(a[i]==b[j])
{
j++;
count++;
}
}
if(count==lens2)
cout<<"\nSubstring"<<endl;
else
cout<<"\nNot Substring"<<endl;
}
```

6.OUTPUT->

```
1: Copy
2:Reverse
3:Substring  0.exit
Select Option:
1
```

```
Enter the stringshivam
Copied String :- shivam

1: Copy
2:Reverse
3:Substring  0.exit
Select Option:
2

Enter string.
abcd
4
Reversed string is: dcba

1: Copy
2:Reverse
3:Substring  0.exit
Select Option:
3

Enter a String :-   shivam
Enter the Substring:-   ivam

Substring
1: Copy
2:Reverse
3:Substring  0.exit
Select Option0
```

7.Conclusion->

d. from this assignment we came to know that how to pass the base address of character array and accept in pointer.

e. We came to know that how the implicate conversion from character to its ASS11 value takes place.
f. Hwo to perform the operations on pointer such as incrementation ,referencing , dereferencing .

Skill development
Assignment – 04

1. **Aim:**

Implement a menu driven Program in C++ for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: EMP-ID, Name, Dept, Designation, PhNo.

a.Create a DLL of N Employees Data.
b. Display the DLL .
c.Perform Insertion.

2. **Objective:**

a .To understand the concept of doubly linked list.
b .To modify the singly linked list into doubly linked list by using the previous pointer.

3. **Theory->**

Arrays can be used to store linear data of similar types, but arrays have following limitations.
1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).
Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

Advantages of linked list:

1) Dynamic memory allocation.
2) Ease of insertion/deletion than arrays.

Representation:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.

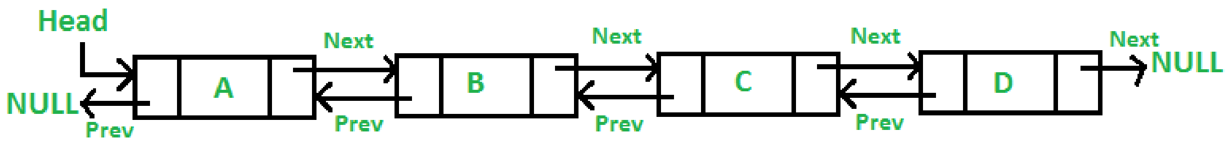
Each node in a list consists of at least two parts:

1) data

2) Pointer (Or Reference) to the next node

In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.



How to create DLL?

Declare a structure with for nodes. In structure there will be 3 member. 1st will contains data and 2nd will contains address of next and 3rd will contain address of previous nodes.

How to display the linked list?

In the linked list we have given address of head i.e starting of LL so we can use while loop to traverse the linked list.

Addition of nodes:

A **D**oubly **L**inked **L**ist (DLL) contains an extra pointer, typically called *previous pointer*, together with next pointer and data which are there in singly linked list.

1) Add a node at the front:
The new node is always added before the head of the given Linked List. And newly added node becomes the new head of DLL. For example if the given Linked List is 10152025 and we add an item 5 at the front, then the Linked List becomes 510152025. Let us call the function that adds at the front of the list is push(). The push() must receive a pointer to the head pointer, because push must change the head pointer to point to the new node

2) Add a node after a given node.:
We are given pointer to a node as prev_node, and the new node is inserted after the given node.

Five of the above steps step process are same as the 5 steps used for inserting after a given node in singly linked list. The two extra steps are needed to change previous pointer of new node and previous pointer of new node's next node.

3) Add a node at the end:
The new node is always added after the last node of the given Linked List. For example if the given DLL is 510152025 and we add an item 30 at the end, then the DLL becomes 51015202530.
Since a Linked List is typically represented by the head of it, we have to traverse the list till end and then change the next of last node to new node.

4. <u>**Algorithm:**</u>

Algorithm for creation of DLL.

A. Insert at the end->

("Insert_end" function definition)

I. accept the address of first node of linked list in pointer' first' of datatype structure.
II. declare the two pointers move, newnode of data type structure.
III. Make move pointing to first.
IV. Accept the data from user in getdata function which returns the base address of newly created node , accept it in newnode.
V. Check if first is equal to NULL.
VI. If it is equal to null.
   a. make newnode as first node .
   b. previous part of newnode pointing to Itself.

VII. If first is not equal to NULL.
   a. Then traverse the linked list from first node to last node.
   b. Insert the newnode at the end of linked list.
   c. Make the previous part of newnode pointing to second last node.
VIII. Return first .

Insert at the beginning ->

("Insert_beg" function definition)

I. accept the address of first node of linked list in pointer first of datatype structure.
II. declare the two pointers move, newnode of data type structure.
III. Make move pointing to first.
IV. Accept the data from user in getdata function which returns the base address of newly created node , accept it in newnode.

V. If it is equal to null.
   a. make newnode as first node .
   b. previous part of newnode pointing to Itself.

VI. If first is not equal to NULL.

   a. Make next part of newnode pointing to first.

```
        b. Previous part of first pointing to newnode.
        c. Make previous part of newnode pointing to NULL.
        d. Equate first equal to newnode.

VII. Return first
```

B. Insert at the given position ->

("Insert_position" function definition)

```
I.    accept the address of first node of linked list in pointer first of
      datatype structure.
II.   declare the two pointers move, newnode of data type structure.
III.  Make move pointing to first.
IV.   Accept the data from user in getdata function which returns the
      baseaddress of newly created node , accept it in newnode.
V.    Accept the position of insertion of node in pos.
VI.   Declare the integer variable count equals to 1.
VII.  If first is equal to null.

          a. make newnode as first node .
          b. previous part of newnode pointing to  Itself.

VIII. If first is not equal to NULL.

          a. Traverse the linked list up to pos-1 .
          b. Make move pointing to move->next.
          c. Increament the count.


          c. Make next parta of newnode pointing to next part of
             move.
          d. Make previous part of newnode pointing to move.
          e. Make previous part of next node pointing to
             newnode.
          f. Make next part of move pointing to newnode.

IX.   Return first.
```

C. Display ->

("display" function defination)

```
I.   accept the address of first node of   list in pointer first of datatype structure.
II.  declare the  pointers move,  of data type structure.
III. Make move pointing to first.
IV.  Traverse the linked list till move is not equal to NULL.
        a. Simultaneously print the data from each node .
        b. Increament the move pointer by pointing to next node.
```

D. Algorithm for structure->

Create a structure having entities

1. Department name
2. Designation
3. Id
4. Ph no
5. Name
6. Next pointer pointing to next node.
7. Prev pointer pointing to previous node.

E. Algorithm for getdata function->

```
I.   Create a pointer "record" of data type structure .
II.  Assign dynamically attributes of structure to it.
III. Accept the data from user .
IV.  Return record.
```

F. Algorithm for main function

```
I.   Eqate both previous and next pointer to NULL.
II.  Show menu driven program for user using switch case.
III. Show output as follows
              1.insert node at end
              2.insert node at position
              3. insert node at begining .
              4.display.

IV.  If user enter 1 then call 'insert_end'  function by passing first as parameter and
     accept the return value from this function in first.
```

V. If user enter 2 then call 'insert_psition' function by passing first as parameter and accept the return value from this function in first.

VI. If user enter 3 then call 'insert_beg' function by passing first as parameter and accept the return value from this function in first.

VII. If user enter 4 then call 'display' function by passing first as parameter .

VIII. Return 0.

5. **Code->**

```
#include<iostream>
#include<string.h>
using namespace std;

typedef struct linked_list
{
int id;
char name[10];
char dep[10];
char no[10];
char posi[10];
struct  linked_list *next;
struct linked_list *prev;
}node;

node*getdata()
{
   cout<<endl;
node *record;
record=new node;
cout<<"enter your name"<<endl;
cin>>record->name;
cout<<"enter the name of your department"<<endl;
cin>>record->dep;
cout<<"enter the  Designation of you"<<endl;
cin>>record->posi;
cout<<"enter the id"<<endl;
```

```
cin>>record->id;
cout<<"enter your contact no"<<endl;
cin>>record->no;

record->next=NULL;
record->prev=NULL;
return record;

}
            // INSERT AT END
node* insert_end(node *head)
{
node *move,*newnode;
newnode=getdata();

if(head==NULL)
{
head=newnode;
   newnode->prev=head;
cout<<"head is created!!!";
}

else
{
move=head;
while(move->next!=NULL)
move=move->next;
move->next=newnode;
newnode->prev=move;
}
return head;
}

               // INSETR AT POSITION
 node *insert_mid(node *head)
 {
  int pos,count=1;
  node *move,*newnode;
```

```
move=head;
cout<<" enter the position at which you want to insert";
cin>>pos;
newnode=getdata();
move=head;
while(count!=pos-1)
 {
 move=move->next;
 count++;
 }
 newnode->next=move->next;
  newnode->prev=move;
(move->next)->prev=newnode;
 move->next=newnode;
  return head;
}

            // INSERT AT BEGINING
node *insert_beg(node *head)
{
node *newnode;
newnode=getdata();
newnode->next=head;
head->prev=newnode;
head=newnode;
newnode->prev=NULL;
return head;
}
          // DISPLAY LINKED LIST
void display(node* head)
{
node *move;
move=head;
while(move!=NULL)
{
cout<<"\n\t"<<"id::      "<<move->id<<"\n\t"<<"name::";
cout<<move->name<<"\n\t"<<"department::  "<<move->dep<<"\n\t"<<
  "contact_no  "<<move->no<<"\n\t"<<"position:  "<<move->posi<<"\n\t";
```

```
cout<<"\n\t";
cout<<"**********";
cout<<"\n\t";
move=move->next;
}
}
int main()
{

    node*head;
    head=NULL;
    int ch;
    do
    {
           cout<<endl<<endl<<endl;
       cout<<"1. create linklist \n 2.insert at position \n3. insert at beginning \n 4.display 0.exit";
        cout<<endl;
       cout<<"enter your choise";

       cin>>ch;
       switch(ch)
       {
       case 1:
        head=insert_end(head);
        break;

         case 2:
        head=insert_mid(head);
        break;

         case 3:
        head=insert_beg(head);
        break;

         case 4:
            display(head);
            break;

        }
cout<<endl<<endl;
    }while(ch!=0);

    return 0;
    cout<<endl<<endl;
}

```

6.Output->

```
1. create linklist

 2.insert at position

3. insert at beginning

 4.display 0.exit

enter your choise1



enter your name

shivam

enter the name of your department

comp

enter the  Designation of you

br

enter the id

11

enter your contact no

9011995001
```

```
head is created!!!

1. create linklist

 2.insert at position

3. insert at beginning

 4.display 0.exit

enter your choise3


enter your name

yash

enter the name of your department

comp

enter the  Designation of you

br

enter the id

10

enter your contact no

9011995002


1. create linklist

 2.insert at position

3. insert at beginning

 4.display 0.exit
```

```
enter your choise2
 enter the position at which you want to insert2


enter your name
shardual
enter the name of your department
comp
enter the  Designation of you
ce
enter the id
102
enter your contact no
9011995003


1. create linklist
 2.insert at position
3. insert at beginning
 4.display 0.exit
enter your choise4


        id::     10
        name::    yash
```

```
    department::  comp

    contact_no  9011995002

    position:: br


    **********

    id::     102

    name::   shardual

    department::  comp

    contact_no  9011995003

    position:: ce
   **********
    id::     11

    name::   shivam

    department::  comp

    contact_no  9011995001

    position::br


    **********


1. create linklist

 2.insert at position

3. insert at beginning
```

```
 4.display 0.exit

enter your choise
```

7.Conclusion->

a. We understood the concept of doubly linked list how it can traverse in both direction.
b. By performing DLL we understood that how linked list save the memory for storage of data .

We conclude that the insertion and deletion is much easy and less time consuming than that of insertion and deletion of array.

Skill development
Assignment – 05

154. **Aim:**

Implement a menu driven Program in C++ for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: EMP-ID, Name, Dept, Designation, PhNo.

a.Create a DLL of N Employees Data.
b. Display the DLL .
c.Perform deletion.

155. **Objective:**

a .To understand the concept of doubly link list.
b .To modify the singly link list into doubly link list by using the previous pointer.

156. **Theory:**

Arrays can be used to store linear data of similar types, but arrays have following limitations.

1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.

2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].
id[] = [1000, 1010, 1050, 2000, 2040].
And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).
Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

Advantages of linked list:

1) Dynamic memory allocation.

2) Ease of insertion/deletion than arrays.

Representation:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.

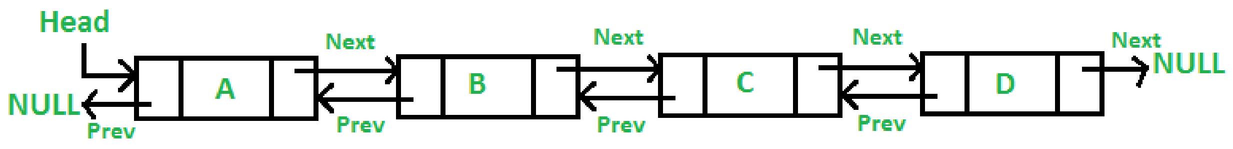
Each node in a list consists of at least two parts:

1) data

2) Pointer (Or Reference) to the next node

In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.



How to create DLL?

Declare a structure with for nodes. In structure there will be 3 member. 1st will contains data and 2nd will contains address of next and 3rd will contain address of previous nodes.

How to display the linked list?

In the linked list we have given address of head i.e starting of LL so we can use while loop to traverse the linked list.

How to delete from linked list?

Given a singly linked list and a position, delete a linked list node at the given position.

Example:

Input: position = 1, Linked List = 8->2->3->1->7

Output: Linked List = 8->3->1->7

Input: position = 0, Linked List = 8->2->3->1->7
Output: Linked List = 2->3->1->7
If node to be deleted is root, simply delete it. To delete a middle node, we must have pointer to the node previous to the node to be deleted. So if positions is not zero, we run a loop position-1 times and get pointer to the previous node.

157. <u>**Algorithm:**</u>

Algorithm for creation of DLL.

G. Create_linklist->

<u>("create_linklist" function definition)</u>

```
IX.   accept the address of first node of link list in pointer' first' of datatype
      structure.
X.    declare the two pointers move, newnode of data type structure.
XI.   Make move pointing to first.
XII.  Accept the data from user in getdata function which returns the base
      address of newly created node , accept it in newnode.
XIII. Check if first is equal to NULL.
XIV.  If it is equal to null.
          c. make newnode as first node .
          d. previous part of newnode pointing to Itself.

XV.   If first is not equal to NULL.
          d. Then traverse the link list from first node to last node.
          e. Insert the newnode at the end of link list.
          f. Make the previous part of newnode pointing to second last node.
Return first .
```

H. Display ->
("display" function defination)

V. accept the address of first node of link list in pointer first of datatype structure.
VI. declare the pointers move, of data type structure.
VII. Make move pointing to first.
VIII. Traverse the linklist till move is not equal to NULL.
  c. Simultaneously print the data from each node .
  d. Increament the move pointer by pointing to next node.

I. Algorithm for structure->

Create a structure having entities

8. Department name
9. Designation
10. Id
11. Ph no
12. Name
13. Next pointer pointing to next node.
14. Prev pointer pointing to previous node.

J. Algorithm for getdata function->

V. Create a pointer "record" of data type structure .
VI. Assign dynamically attributes of structure to it.
VII. Accept the data from user .
VIII. Return record.

K. Algorithm for main function->

IX. Eqate both previous and next pointer to NULL.
X. Show menu driven program for user using switch case.
XI. Show output as follows

```
1.create linklist
2.delete node at position
3. delete node at begining .
4.display.
```

XII. If user enter 1 then call 'create_linklist' function by passing first as parameter and accept the return value from this function in first.

XIII. If user enter 2 then call 'delete_psition' function by passing first as parameter and accept the return value from this function in first.

XIV. If user enter 3 then call 'delete_beg' function by passing first as parameter and accept the return value from this function in first.

XV. If user enter 4 then call 'display' function by passing first as parameter .

XVI. Return 0.

L. Algorithm for deletion->

1. Delete at end:

```
X.    accept the address of first node of link list in pointer first of
      datatype structure.
XI.   declare the two pointers moveof data type structure.
XII.  Make move pointing to first.
XIII. Check if first equals to NULL
          a. Then give output as linklist is empty

XIV.  If first is not equal to null.
          a. Traverse the linklist up to last node .
          b. By using move pointer and incrementing it in every
             successful check .
          c. Make the next part of node previous to move
             pointing to NULL.
          d. Make previous part of move pointing to NULL.
          e. display the data of node to be deleted node.
          f. Delete the node.
XV.   Return first.
```

2. Delete at beginning:

```
I.   accept the address of first node of link list in pointer first of datatype
     structure.
II.  declare the two pointers moveof data type structure.
III. Make move pointing to first.
IV.  Check if first equals to NULL
         a.Then give output as linklist is empty
```

```
V. If first is not equal to null.
      a. Make first pointing to next part of move.
      b. Make previous part of head pointing to NULL.
      c. Make next part of move pointing to NULL.
VI. Return first.
```

3. **Delete at position:**

```
I. accept the address of first node of link list in
   pointer first of datatype structure.
II. declare the two pointers moveof data type structure.
III. Decleare tow integer variables 'count 'having intial value 1 and
     'pos'.
IV. Accept the position of deletion from user.
V. Make move pointing to first.
VI. Check if first equals to NULL
      a.Then give output as linklist is empty

VII. If first is not equal to null.
      a. Traverse the linklist up to last node .
      b. By using move pointer and count ;
      c. By incrementing move and count in every
         successful check .
      d. Make previous part of  node which is  next to move
         pointing to  node which  is previous to move.
      e. Make next part of node which is to move previous
         pointing to node which is  next to move.
      f. Make next part of move pointing to NULL.
      g. Make previous part of move pointing to NULL.
      h. Display the content of node which is to be deleted.
      i. Delete the move pointer.
VIII. Return first.
```

5.Code->

```
#include<iostream>
#include<string.h>
using namespace std;

typedef struct linklist
{
int id;
char name[10];
char dep[10];
char no[10];
char posi[10];
struct linklist *next;
struct linklist *prev;
}node;

node*getdata()
{
node *record;
record=new node;
cout<<"enter your name"<< "\t";
cin>>record->name;
cout<<"enter the name of your department"<< "\t";
cin>>record->dep;
cout<<"enter the  Designation of you"<< "\t";
cin>>record->posi;
cout<<"enter the id"<< "\t";
cin>>record->id;
cout<<"enter your contact no"<< "\t";
cin>>record->no;

record->next=NULL;
record->prev=NULL;
return record;
```

```
}
            // createlinklist
node* create_linklist(node *head)
{
node *move,*newnode;
newnode=getdata();

if(head==NULL)
{
head=newnode;
    newnode->prev=head;
cout<<"head is created!!!";
}

else
{
move=head;
while(move->next!=NULL)
     move=move->next;

move->next=newnode;
newnode->prev=move;
}
return head;

}

                // DISPLAY LINKLIST
void display(node* head)
{
node *move;
move=head;
while(move!=NULL)
{
```

```
cout<<"\n\t"<<"id::      "<<move->id<<"\n\t"<<"name::";
cout<<move->name<<"\n\t"<<"department::  "<<move->dep<<"\n\t"<< "contact_no
"<<move->no<<"\n\t"<<"position:  "<<move->posi<<"\n\t";
cout<<"\n\t";
cout<<"**********";
cout<<"\n\t";

move=move->next;
}
}
        // delete at beginning
node *delete_beg(node *head)
{
node *move;
move=head;
head=head->next;

head->prev=NULL;
move->next=NULL;

cout<<"deleted record is";
cout<<"\n\t"<<"id::      "<<move->id<<"\n\t"<<"name::";
cout<<move->name<<"\n\t"<<"department::  "<<move->dep<<"\n\t"<< "contact_no
"<<move->no<<"\n\t"<<"position:  "<<move->posi<<"\n\t";
cout<<"\n\t";
cout<<"**********";
cout<<"\n\t";
delete move;
return head;
    }
          // DELETE AT END
node *delete_end(node *head)
{
 node *move;
  move=head;
```

```
if(head==NULL)
 {
cout<<"list is empty";
 }
 else
 {
  while(move->next!=NULL)
   {
    move=move->next;
    }(move->prev)->next=NULL;
move->prev=NULL;

    cout<<"the deleted record is";
  cout<<"\n\t"<<"id::      "<<move->id<<"\n\t"<<"name::";
cout<<move->name<<"\n\t"<<"department::  "<<move->dep<<"\n\t"<< "contact_no
"<<move->no<<"\n\t"<<"position:  "<<move->posi<<"\n\t";
cout<<"\n\t";
cout<<"**********";
cout<<"\n\t";

   delete move;
  }
  return head;
}
          // delete at given position
node * Delete_position(node *head)
{
int pos,count=1;
cout<<"enter the which you want to delete ";
cin>>pos;

node*move;
move=head;
while(count!=pos)
{
```

```
move=move->next;
count++;
}
(move->next)->prev=move->prev;
(move->prev)->next=move->next;


move->next=NULL;
move->prev=NULL;
cout<<"deleted record is";
cout<<"\n\t"<<"id::      "<<move->id<<"\n\t"<<"name::";
cout<<move->name<<"\n\t"<<"department::  "<<move->dep<<"\n\t"<< "contact_no
"<<move->no<<"\n\t"<<"position:  "<<move->posi<<"\n\t";
cout<<"\n\t";
cout<<"**********";
cout<<"\n\t";
delete move;
return head;
}
int main()
{

    node*head;
    head=NULL;
    int ch;
    do
    {
           cout<<endl<<endl<<endl;
       cout<<"1. create linklist \n 2.delete at position \n3. delete at beginning
\n4.delete at end\n 5.display \n 0.exit";
        cout<<endl;
       cout<<"enter your choise";

       cin>>ch;
       switch(ch)
```

```
        {
        case 1:
         head=create_linklist(head);
         break;
          case 2:
         head=Delete_position(head);
         break;

          case 3:
         head=delete_beg(head);
         break;
          case 5:
             display(head);
             break;
            case 4:
         head=delete_end(head);
         break;
       }
   }while(ch!=0);
              Return 0;
 }
```

6.Output->

```
1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise1
enter your name shivam
enter the name of your department       comp
enter the  Designation of you   br
enter the id    11
enter your contact no   9011995001
```

```
head is created!!!



1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise1
enter your name yash
enter the name of your department       comp
enter the  Designation of you   br
enter the id    12
enter your contact no   9011995002


1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise1
enter your name shardual
enter the name of your department       comp
enter the  Designation of you   cr
enter the id    13
enter your contact no   9011995003


1. create linklist
 2.delete at position
3. delete at beginning
```

```
4.delete at end
 5.display
 0.exit
enter your choise1
enter your name kartik
enter the name of your department       comp
enter the  Designation of you   br
enter the id    14
enter your contact no   9011995004




1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise5

        id::      11
        name::shivam
        department::  comp
        contact_no  9011995001
        position: br

        **********


        id::      12
        name::yash
        department::  comp
        contact_no  9011995002
        position: br
```

```
          **********

          id::      13
          name::shardual
          department::  comp
          contact_no  9011995003
          position: cr

          **********

          id::      14
          name::kartik
          department::  comp
          contact_no  9011995004
          position: br

          **********




1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise3
deleted record is
          id::      11
          name::shivam
          department::  comp
          contact_no  9011995001
          position: br
```

```
        * * * * * * * * * *




1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise4
the deleted record is
        id::      14
        name::kartik
        department::  comp
        contact_no  9011995004
        position: br

        * * * * * * * * * *




1. create linklist
 2.delete at position
3. delete at beginning
4.delete at end
 5.display
 0.exit
enter your choise2
enter the which you want to delete 2
the deleted record is
        id::      13
        name::shardual
        department::  comp
        contact_no  9011995003
        position: cr

Press any key to continue.
```

7.Conclusion->

c. We understood the concept of doubly linklist how it can traverse in both direction.
d. By performing DLL we understood that how linklist save the memory for storage of data .
e. We conclude that the insertion and deletion is much easy and less time consuming than that of insertion and deletion of array.

Skill development

Assignment – 06

I. Aim:

Develop and Implement a Program in C++ for converting an Infix Expression to Postfix Expression.

II. Objective:

to develop a c++ program for converting infix to postfix expression .Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %,^ and alphanumeric operands.

III. Theory->

1) Infix notation: X + Y
   Operators are written in-between their operands. This is the usual way we write expressions. An expression such as A * (B + C) / D is usually taken to mean something like: "First add B and C together, then multiply the result by A, then divide by D to give the final answer."

   Infix notation needs extra information to make the order of evaluation of the operators clear: rules built into the language about operator precedence and associativity, and brackets ( ) to allow users to override these rules. For example, the usual rules for associativity say that we perform operations from left to right, so the multiplication by A is assumed to come before the division by D. Similarly, the usual rules for precedence say that we perform multiplication and division before we perform addition and subtraction.

2) Postfix notation ->
   (also known as "Reverse Polish notation"): X Y +

   Operators are written after their operands. The infix expression given above is equivalent to A B C + * D /
   The order of evaluation of operators is always left-to-right, and brackets cannot be used to change this order. Because the "+" is to the left of the "*" in the example above, the addition must be performed before the multiplication.
   Operators act on values immediately to the left of them. For example, the "+" above uses the "B" and "C". We can add (totally unnecessary) brackets to make

this explicit:
((A (B C +) *) D /)
Thus, the "*" uses the two values immediately preceding: "A", and the result of the addition. Similarly, the "/" uses the result of the multiplication and the "D".

IV. Algorithm:

- Read expression from left to right.
- If ch (i.e. current symbol in input string) is operand then print it in output string
- If ch is '(' then push it onto the stack
- If ch is a first operator or first operator after '(' then push it onto the stack.
- If priority of ch <= that of element on the top of stack, then pop the top element & push the ch onto the stack.
- If priority of ch > that of top element then push ch on the stack
- At the end of input string pop all entries from the stack till it becomes empty & write them into the output string.
- If ch is a ') then pop all the entries from the stack and add them into the output string till you encounter '('

V. Code:

```
#include<iostream>

#define SIZE 10

using namespace std;

class stack

{

 char data[SIZE];

 int top;
 public:
  stack();
  void push(char);
  char pop();
  int isempty();
  char peek();
};
stack::stack()
{
 top=-1;
}
void stack::push(char ch)
{
 top++;
 data[top]=ch;
}
char stack::pop()
{
 int no;
 no=data[top];
 top--;
 return no;
}
```

```
int stack::isempty()
{
 if(top==-1)
 {
  return 1;
 }
 return 0;
}
char stack::peek()
{
 return data[top];
}
int priority(char ch)
{
 if(ch=='^'||ch=='$')
 {
  return 3;
 }
 if(ch=='/'||ch=='*'||ch=='%')
 {
  return 2;
 }
 if(ch=='+'||ch=='-')
```

```
  {
   return 1;
  }
  return 0;
 }
 int main()
 {
  char infix[20],postfix[20],ch;
  int  i,j=0;
  stack s;
  cout<<"\nEnter the infix expression=>";
  cin>>infix;
  for(i=0;infix[i]!='\0';i++)
  {
   switch(infix[i])
   {
    case '(':
     s.push('(');
     break;
    case '$':
    case '^':
    case '*':
    case '/':
```

```
 case '%':
 case '+':
 case '-':
  while(!s.isempty()&&(priority(s.peek()))>=priority(infix[i]))
  {
   postfix[j]=s.pop();
   j++;
  }
  s.push(infix[i]);
  break;
 case ')':
  ch=s.pop();
  while(ch!='(')
  {
   postfix[j]=ch;
   ch=s.pop();
   j++;
  }
  break;
 default:
  postfix[j]=infix[i];
  j++;
}
```

```
}
while(!s.isempty())
{
 postfix[j]=s.pop();
 j++;
}
postfix[j]='\0';
cout<<"\nPostfix string is =>"<<postfix;
return 0; }
```

VI. Output->

```
Enter the infix expression=>a+b*c-d/e*f


Postfix string is =>abc*+de/f*-
Process returned 0 (0x0)   execution time : 44.937 s
Press any key to continue.
```

VII. Conclusion->

From this assignment we came to know that how the stack is being used to convert the infix expression to postfix expression.
The push and pop operation related to stack are used .
This is an example of use of stack.

Skill development

Assignment – 07

I. **AIM :-**

Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

II. **Objective:**

Implement the pizza parlor management system using circular queue.
Perform the insertion at beginning and deletion at end.

III. **Theory->**

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **'Ring Buffer'**.

- **Front:** Get the front item from queue.
- **Rear:** Get the last item from queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.

IV. **Algorithm:-**

```
1. Start.
2. Create a structure for the queue, with one array of size M and two variables which act as pointer to the front and rear of the queue.

  A. Function to check queue is full
        a) If front and rear point to the same location then queue is empty.
        b) If (rear%M)+1=front then the queue is full.

  B. 4. function to add element in queue
        a) Check if the queue is full or not.
```

b) If it is not full then add element to the rear end of the queue.
c) Rear=(rear%M)+1.

C. function to delete element in queue
   a) Check if the queue is empty or not.
   b) If it is not empty then remove element from the front.
   c) Front=(front%M)+1.

D. Function to display elements in queue
   a) Take a temporary pointer which points to front.
   b) Keep displaying elements of the queue as long as the temporary pointer does not reach rear of the queue.

E. in main function
1. Take two variables to strcture. Start and rear.
2. Make them pointing to -1.
3. Show the menu driven program for user
   1.insert
   2.delete
   3. display

1. If user enter 1 then call the enqueue function.
2. If user enter 2then call the dequeue function.
3. If user enter 3 then call the display function.
4. If user enter 4 then terminate the program.

8. Take input from the user, and perform the respective function.

V. Code->

```
#include <iostream>
 using namespace std;
 const int M=50;
 struct queue {
   int q[M],rear, start;
 }qe;
 int qfull() {
  if(((qe.rear%M)+1)==qe.start)
   return 1;
    else
    return 0;
}
 int qempty()
{     if(qe.start==qe.rear)
     return 1;
    else
 return 0;
 }
 void enqueue(int data)
{     if(qfull())
    {         cout<<"Queue ids full, please wait\n";
   }
 else
 {
    qe.q[qe.rear]=data;
    qe.rear=(qe.rear%M)+1;
  }
 }
 Void  dequeue()
 {
   if(qempty())
 {
 cout<<"Queue is empty, please insert data first\n";
  }
  else
    {
  int data=qe.q[qe.start];
```

```
    qe.start=(qe.start%M)+1;
cout<<" take your order "<<data<<" pizzas have been delivered.\n";
  }
}
 void display()
 {
if(qempty())
       cout<<"The queue is empty, please give orders first\n";
   else
    {
     int i=qe.start;
       while(((i%M)+1)!=qe.rear)
     {
   cout<<qe.q[i]<<endl;
      i=(i%M)+1;
      }
  cout<<qe.q[i]<<endl;
  }
}
 int main()
 {
   qe.start=-1
;
    qe.start=qe.rear;
     char op;
  do
   {
     int c;
        cout<<"1] Add order\n2] Delete order\n3] Display list of orders\n 0.exit";
Enter your choice ";
    cin>>c;
       switch(c)
        {
      case 1: {
              int data;
```

```
            cout<<" give order of pizza  ";
                cin>>data;
         enqueue(data);
       }
           break;
       case 2:
{
                        dequeue();

       }
    break;
           case 3:
 display();
                break;
      default:cout<<"Process unavailable. Please try again.\n";
      }
 }while(c!=0);
 return 0;
 }
```

VI. Output->

```
1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 1
give order of pizza  2


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 1
```

```
give order of pizza 3


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 1
give order of pizza 4


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 3
2
3
4


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 2
take your order. 2 pizzas have been delivered.


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 2
take your order. 3 pizzas have been delivered.
```

```
1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 2
take your order. 4 pizzas have been delivered.


1] Add order
2] Delete order
3] Display list of orders
 0.exit
Enter your choice 2
Queue is empty, please give orders first
```

VII. Conclusion->

From this assignment we understood the insertion and deletion operation in circular queue.
The data is overrriden when we insert data and the queue is full.
The insertion is done at end.
The deletion is done at beginning.

Skill development

Assignment 8

I. Aim –

a) Sort the set of strings in ascending order using Bubble sort and descending order by using Selection sort (Display pass by pass output)
b) Search a number using Fibonacci search.

II. Objective –

Use of bubble and selection sort.

III. Theory –

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list, compares adjacent pairs and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and impractical for most problems even when compared to insertion sort.

In computer science, selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n^2) time complexity, making it inefficient on large lists, ...
Worst-case space complexity: O(1) auxiliary
Worst-case performance: O(n2) comparisons, ...
Best-case performance: O(n2) comparisons, O...
Average performance: O(n2) comparisons, O(n)

IV. Algorithm –

Step – 1 : Start.

Step – 2 : Define str and n.

Step – 3 : Write ascend function which includes x=0 , and n.

Use for loop to increased passes.

Use temp char to compare two characters of string.
And display the final result.
Step – 4 : Write descend function which includes x=0 and passes=0.
Again use the character I and min.
Compare the strings and display the output.
Step – 5 : Now declare int i , key , f .
Now enter total number of elements.
Enter the element to be search.
Use search function to check weather the entered element is present or not.
Step – 5 : Stop.

V. CODE–

```
#include<iostream>
#include<string.h>
using namespace std;

char str[10];
int n;

void ascend()
{
int x=0;
int passes=0;
for(int i=0;i<n-1;i++)
{
passes++;
char temp;
for(int j=0;j<n-1;j++)
{
if(str[j]>str[j+1])
{
temp=str[j];
str[j]=str[j+1];
str[j+1]=temp;
x++;
}
}
cout<<"Pass "<<passes<<" :\t";
if(x==0)
{
cout<<"Already sorted\n";
}
else
cout<<"The sorted sequence is\t"<<str<<"\n";
}

}

void descend()
{
int x=0,j;
int passes=0;
for(int i=0;i<n-1;i++)
{
passes++;
char temp;
char min=i;
for(j=i+1; j<n;j++)
{
if(str[j]>str[min])
{
temp=str[j];
str[j]=str[min];
str[min]=temp;
x++;
}
```

```
}
cout<<"Pass "<<passes<<" :\t";
if(x==0)
{
cout<<"Already sorted\n";
}
else
cout<<"The sorted sequence is\t"<<str<<"\n";
}
}

int main()
{
int choice;
do {
cout<<"Enter characters you want to sort\n";
cin>>str;
n=strlen(str);
cout<<"Enter \n1 to sort in ascending order\n2 to sort in descending order\n3 to
exit\n";
cin>>choice;
switch(choice)
{
case 1:
ascend();
break;
case 2:
descend();
break;
case 3:
cout<<"EXIT\n";
break;
default:
```

```
cout<<"Invalid input\n";
}
}while(choice==3);
return 0;
}
```

b)

```
#include<iostream>
using namespace std;
void search(int arr[],int n,int key,int f,int b,int a);
int fib(int n);
int main()
{
int arr[20];
int n,key,f;
int i;
cout<<"Enter the total no.of element"<<endl;
cin>>n;
cout<<"Enter the number";
for(i=1;i<=n;i++)
cin>>arr[i];

cout<<"Enter the elements to be search"<<endl;
cin>>key;
search(arr,n,key,n,fib(n),fib(fib(n)));
return 0;
}

void search(int arr[],int n,int key,int f,int b,int a)
{
if(f<1||f>n)
cout<<"The number is not present";
else if(key<arr[f])
{
if(a<=0)
cout<<"The element is not present "<<endl;
else
search(arr,n,key,f-a,a,b-a);
}
else if(key>arr[f])
{
if(b<=1)
cout<<"The element is not present"<<endl;
else
search(arr,n,key,f+a,b-a,a-b);
}
else
cout<<"The element is present at "<<f<<endl;
}

int fib(int n)
{
int a,b,f;
if(n<1)
return n;
a=0;b=1;
while(b<n)
{
f=a+b;
a=b;
b=f;
}
return a;
}
```

VI. Output –

A)

Case 1 –

```
Enter characters you want to sort
sanket
Enter
1 to sort in ascending order
2 to sort in descending order
3 to exit
1
Pass 1 :    The sorted sequence is  ankest
Pass 2 :    The sorted sequence is  akenst
Pass 3 :    The sorted sequence is  aeknst
Pass 4 :    The sorted sequence is  aeknst
Pass 5 :    The sorted sequence is  aeknst
```

Case 2 –

```
Enter characters you want to sort
sanket
Enter
1 to sort in ascending order
2 to sort in descending order
3 to exit
2
Pass 1 :    The sorted sequence is  tankes
Pass 2 :    The sorted sequence is  tsaken
Pass 3 :    The sorted sequence is  tsnaek
Pass 4 :    The sorted sequence is  tsnkae
Pass 5 :    The sorted sequence is  tsnkea
```

B)

```
Enter the total no.of element
4
Enter the number
10
20
```

```
30
40
Enter the elements to be search
20
The element is present at 2.
```

VII. CONCLUSION

The sorting is done by bubble sort in assending order.
The selection sort is to sort in dicending order.
The fabonica search is used to search but the condition is that the array is in assending oeder.

Skill development

Assignment – 09

I. Aim

Write C++ program to store names and mobile numbers of your friends in sorted order on names. a) Search your friend from list using binary search (recursive and non recursive)

II. Objective

Perform the sorting and searching operation on string data type.
Use the recursive and non recursive binary search.

III. Theory

Write C++ program to store names and mobile numbers of your friends in sorted order on names. a) Search your friend from list using binary search (recursive and non recursive)

1. Compare x with the middle element.
2. If x matches with middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
4. Else (x is smaller) recur for the left half.

Condition for binary search->

The given set must obey the following rules if we want to apply binary search –

1. All pairs of elements in the given set must be comparable. Mathematically, this is called a totally ordered set.
2. The given set must be sorted, in ascending or descending order.

To sort the elements in assending order we use the and sorting method.
In this example we use the bubble sort

Bubble sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

IV. Algorithm

a. Algorithm for Structure

Declare a structure having entity
name , roll, phone no,and two temporary character array .

b. Algorithm for sorting

Use two for loops

Outer loop is qual to no of variables -1.

The inner for loop for compairing the each element with its successive.

Compare the two strings if they are not in sorted order then swap both of them.

In similar way swap the roll no and phone no with each other.

After sorting perform the binary search which search the name in given list.

c. algorithm for binary search

declare two variables mid, lower ,upper .

initially the lower =0,

upper =size -1.

Mid=(lower+upper)/2.

1. Start with mid element
   - If the **target** value is equal to the middle element of the array, then return the index of the middle element.
   - If not, then compare the middle element with the target value,

- If the target value is greater than the number in the middle index, then pick the elements to the right of the middle index, and start with Step 1.
- If the target value is less than the number in the middle index, then pick the elements to the left of the middle index, and start with Step 1.

2. When a match is found, return the index of the element matched.
3. If no match is found, then return `-1`

c.In main function

Display the sorted list .

And display the result if the name is found in list.

V. Code->

```
#include<iostream>
#include<string.h>
using namespace std;
typedef struct contact
{
  char name[20];
  int roll;
  char phone[20];
 char t[20], m[20];
 int temp;
}con;

int binarySearch( con c1[], int ,int ,int);

int main()
{
    int i,j,no ;
          cout<<"enter the no of your contacts";
```

```
        cin>>no;

    con c[no];
        for(i=0;i<no;i++)
        {
                cout<<"\nEnter the "<<i+1<<"st name ::";
                cin>>c[i].name;
                cout<<"Enter phone number ::";
                cin>>c[i].phone;
                cout<<" enter the roll no";
                cin>>c[i].roll;
        cout<<endl;

        }
        cout<<"\nNames in order that you insert ::";
        for(i=0;i<no;i++)
        {
                cout<<"\n"<<c[i].name<<"\t-\t"<<c[i].phone<<"\t-"<<c[i].roll;
        }

        for(i=1;i<no;i++)
        {
                for(j=1;j<no;j++)
                {
                        if(strcmp(c[j-1].name,c[j].name)>0)
                        {
                                strcpy(c[j].t, c[j-1].name);
                                strcpy(c[j-1].name,c[j].name);
                                strcpy(c[j].name,c[j].t);

                                strcpy(c [j].m , c[j-1].phone);
                                strcpy(c[j-1].phone,c[j].phone);
                                strcpy(c[j].phone,c[j].m);

                                c[j].temp=c[j-1].roll;
```

```
                              c[j-1].roll=c[j].roll;
                              c[j].roll=c[j].temp
                         }
                    }
          }
          cout<<"\nNames in alphabetical order ::";
          for(i=0;i<no;i++)
          {
                    cout<<"\n"<< c[i].name<<"\t"<<c[i].phone<<"\t"<<c[i].roll;
          }
cout<<endl;

          int id;
          cout<<"enter the roll no of your friend that you want to search";
          cin>>id;

int result = binarySearch(c, 0, no-1, id);
  if (result == -1)
    cout<<"Element is not present in array"<<endl;
               else
               cout<<"Element is present at index "<<result+1;;
   return 0;
}

int binarySearch( con c1[], int l, int r, int x)

{

   if (r >= l)

   {

        int mid = l + (r - l)/2;

        if (c1[mid].roll == x)

            return mid;

        if (c1[mid].roll > x)
```

```
          return binarySearch(c1, l, mid-1, x);


        return binarySearch(c1, mid+1, r, x);

   }

   return -1;}
```

VI. <u>output</u>

```
Enter the 1st name ::shivam
Enter phone number ::12345
 enter the roll no13


Enter the 2st name ::yash
Enter phone number ::45678
 enter the roll no14


Enter the 3st name ::darshan
Enter phone number ::67890
 enter the roll no12


Enter the 4st name ::ankit
Enter phone number ::56743
 enter the roll no11


Names in order that you insert ::
shivam  -       12345   -13
yash    -       45678   -14
darshan -       67890   -12
ankit   -       56743   -11
Names in alphabetical order ::
```

```
ankit   56743   11
darshan 67890   12
shivam  12345   13
yash    45678   14

enter the roll no of your friend that you want to search13
Element is present at index3
Process returned 0 (0x0)   execution time : 47.124 s
Press any key to continue.
```

VII. Conclusion ->

From this assignment we understood the application of structure.
How the object of class is passed to function and how it is accepted.
The bubble sort and binary search is performed on character array.
The function decleration plays an impotant role in this code.