Practical No:1

Title: Write a Python Program to find and display the set theory Operations.

Problem Statement: In second year computer engineering class, group A student's play cricket, group B students play badminton and group C students play football. Write a Python program using functions to compute following: -
a) List of students who play both cricket and badminton
b) List of students who play either cricket or badminton but not both
c) Number of students who play neither cricket nor badminton
d) Number of students who play cricket and football but not badminton.

(**Note**- While realizing the set duplicate entries are to avoided)

Objective:

- Understand the concept of linear data structure "ARRAY" and how to declare & Use it.
- Understand the basic operation on Array such as Traversing, Insertion, Merging, Searching, Etc.

Outcomes:

- Understood how to use array concept in programming.
- Understood the concept of linear Data Structure.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Theory: In general as we go through the problem statement, we have to find union, intersection, difference & symmetric difference between two sets

I. Union:

The Union of two sets A & B in the set of all those elements that are present in set A or in set B inboth

e.g. A={1,2}B= {1, 2,5}

A U B = {1, 2, 5}.

II. Intersection :

The intersection of two sets A &B is the set of all those elements that are in set A &setB. i.e. Common elements of set A & set B

e.g. A={1,2} B= {1, 2,3}

A ∩ B = {1, 2}

III. Difference:

The elements that are presented in first set but not in second the set is called difference of first set from secondset.

e.g. A={ 1,2,3,4}B={2,3,5}
A-B ={1,4}

IV. Symmetric Difference:

The Symmetric Difference of the two sets A & B denoted by A∆B isdefined.

$$A \triangle B = (A \cup B) - (A \cap B)$$

e.g.A= {a, b,e,g,) B ={d,e,f,g}

$A \triangle B$ = {a,b,d,f}.

Algorithm:

Algorithm Union (arr1 [], arr2 []):
For union of two arrays, follow the following merge procedure.

1) Use two index variables i and j, initial values i = 0, j = 0

2) If arr1[i] is smaller than arr2[j] then print arr1[i] and increment i.

3) If arr1[i] is greater than arr2[j] then print arr2[j] and increment j.

4) If both are same then print any of them and increment both i and j.

5)Print remaining elements of the larger array.

Algorithm Intersection(arr1[], arr2[]):

For Intersection of two arrays, print the element only if the element is present in both arrays.

1) Use two index variables i and j, initial values i = 0, j = 0

2) If arr1[i] is smaller than arr2[j] then increment i.

3) If arr1[i] is greater than arr2[j] then increment j.

4) If both are same then print any of them and increment both i and j.

Pseudo code for union of two sets:

```
# Set union method
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# use | operator
# Output: {1, 2, 3, 4, 5, 6, 7, 8}
print(A | B)
```

Pseudo code for intersection of two sets :

```
# Intersection of sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# use& operator
# Output: {4, 5}
print(A & B)
```

Pseudo code for difference of two sets:

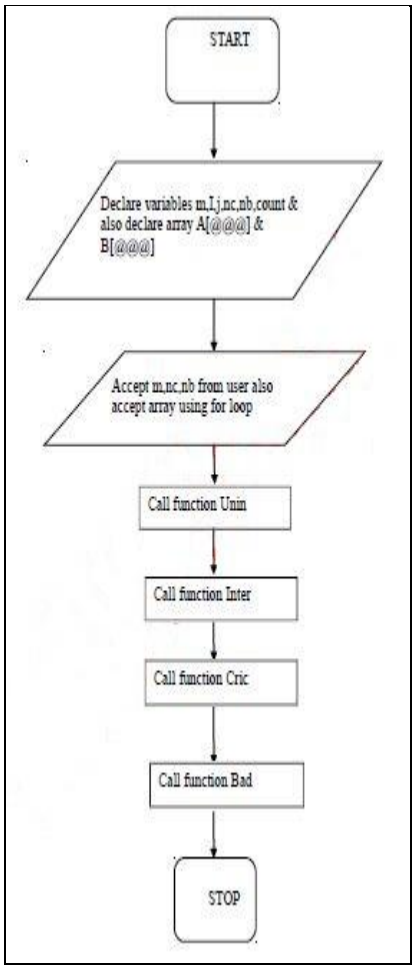
```
# Difference of two sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# use - operator on A
```

```
# Output: {1, 2, 3}
print(A - B)
```

Pseudo code for symmetric difference of two sets :

```
# Symmetric difference of two sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# use ^ operator
# Output: {1, 2, 3, 6, 7, 8}
print(A ^ B)
```

Flowchart:



Conclusion/Analysis:

Understand the concept of Array and able to implement this in Set theory Operations.

Viva Questions:

1) What is an array? Explain array as anADT?
2) Explain polynomial representation using anarray?
3) Explain an address calculation formula & memory representation for one dimensionalarray?
4) What is the difference between array and linklist?

Practical No: 2

Title: Write a Python program to deposit and withdraw money from bank account.

Problem Statement: Write a Python program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following: D 100 W 200 (Withdrawal is not allowed if balance is going negative. Write functions for withdraw and deposit) D means deposit while W means withdrawal.
Suppose the following input is supplied to the program:
D 300, D 300 , W 200, D 100 Then, the output should be: 500

Objective:

- Understand the concept of linear data structure "ARRAY" and how to declare & Use it.
- Understand the concept of different functions & how to declare ,define & call it.

Outcomes:

- Understood how to use array concept in programming.
- Understood the concept of functions.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Theory: In general as we go through the problem statement, we have to do 2 transactions deposit & withdrawl.

I. Deposit

In deposit function we add deposited amount to current balance

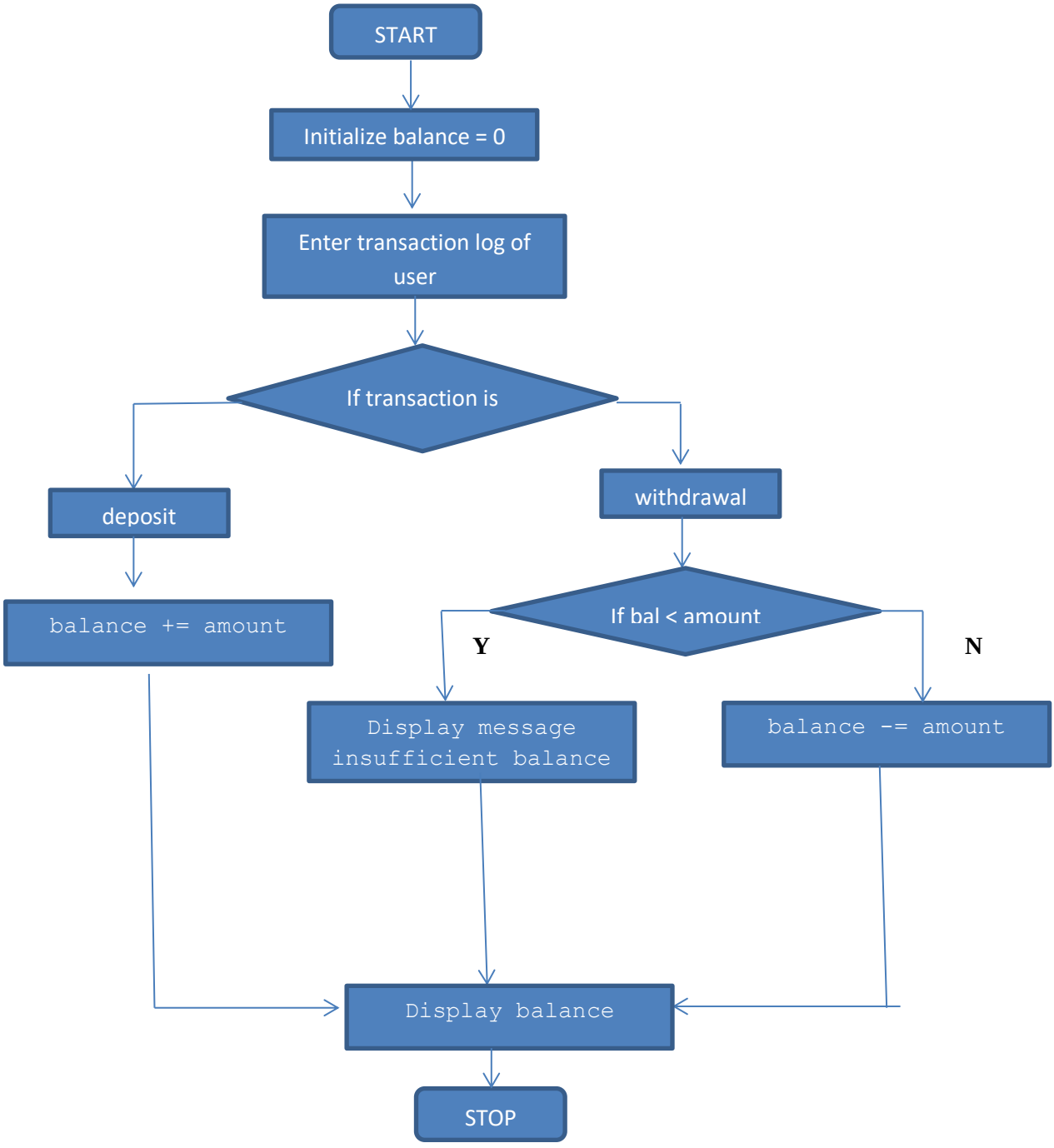
```
def deposit(amount,balance):
  balance += amount
  return balance
```

II. Withdrawl

In withdrawal function we deduct amount from current balance. Withdrawal is not allowed if balance is going negative .

```
def  withdrawal(amount,balance):
  balance = balance-amount
  return balance
```

Flowchart:



Algorithm:

Algorithm Deposit (amount, balance):
For deposit operation , follow the following procedure.

1) Enter the amount to be deposited.
2) Add entered amount in current balance
3) Display updated balance

Algorithm withdrawal (amount, balance):
For deposit operation, follow the following procedure.

1) Enter the amount to be withdrawn.
2) Check if entered amount is greater than balance then display message insufficient balance(Withdrawal is not allowed if balance is going negative)
3) Else subtract entered amount from current balance
4) Display Updated balance.

Conclusion/Analysis:

Understand the concept of Array and functions and able to implement it.

Viva Questions:

1) What is an array?
2) What is function?
3) How to declare,define and call functios.
4) How to pass parameters to function.

Practical No: 3

Title: Write a Python program to to perform various string operations.

Problem Statement: Write a Python program to compute following operations on String:

a) To display word with the longest length
b) To determines the frequency of occurrence of particular character in the string
c) To check whether given string is palindrome or not
d) To display index of first appearance of the substring
e) To count the occurrences of each word in a given string

Objective:

- Understand the concept of String.
- Understand the various operations on string.

Outcomes:

1. Learning & implementations of string operations such as Copy, Length, Reversing, Palindrome, and Concatenation and to find occurrence sub string etc.
2. To understand the use standard library functions for stirng opeations.
3. To perform the string operations.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Theory:

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable.

Phython provides various inbuilt functions for string.

Table of Python String Methods

Function Name	Description
capitalize()	Converts the first character of the string to a capital (uppercase) letter
center()	Pad the string with the specified character.
count()	Returns the number of occurrences of a substring in the string
encode()	Encodes strings with the specified encoded scheme
ljust()	Left aligns the string according to the width specified
lower()	Converts all uppercase characters in a string into lowercase
lstrip()	Returns the string with leading characters removed
maketrans()	Returns a translation table
partition()	Splits the string at the first occurrence of the separator

Conclusion/Analysis:

Understand the concept of String and functions and able to implement on it.

Viva Questions:

1) What is string?
2) What are different function can be performed on string?

Practical No: 4 (A)

Title: Write a python program to search roll numbers of student in array.

Problem Statement: Write a python program to store roll numbers of student in array who attended training program in random order. Write function for searching whether particular student attended training program or not, using Linear search and Sentinel search. Please make note that while realizing the set duplicate entries are to avoided.

Objective:

- Understand the concept of linear data structure "ARRAY" and how to declare & Use it.
- Understand the basic concepts of searching, types of searching and implementations of searching.

Outcomes:

- Use algorithms on various linear data structure using sequential organization to solve real life problems.
- Understood how to implement Linear Search.
- Understood how to implement Sentinel search.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Theory:

- **Searching** is the process of finding a given value position in a list of values. It decides whether a search key is present in the data or not. It is the algorithmic process of finding a particular item in a collection of items.
- Different Searching techniques are,
    1. Linear Search.
    2. Binary Search.
    3. Fibonacci Search.

Linear Search: A linear search or **sequential search** is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched. Time Complexity is O(n).

Algorithm of Linear Search:

Linear Search (Array A, Value x)

Step 1: Set i to 1

Step 2: if i > n then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

Linear Search Pseudo code:

```
procedure linear_search (list, value)
for each item in the list
                                        if match item == value
                                        return the item's location
                                        end if
 end for
end procedure
```

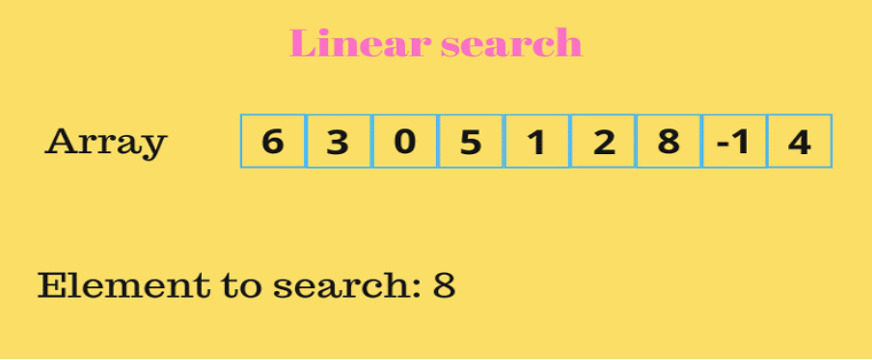


Figure: Linear Search

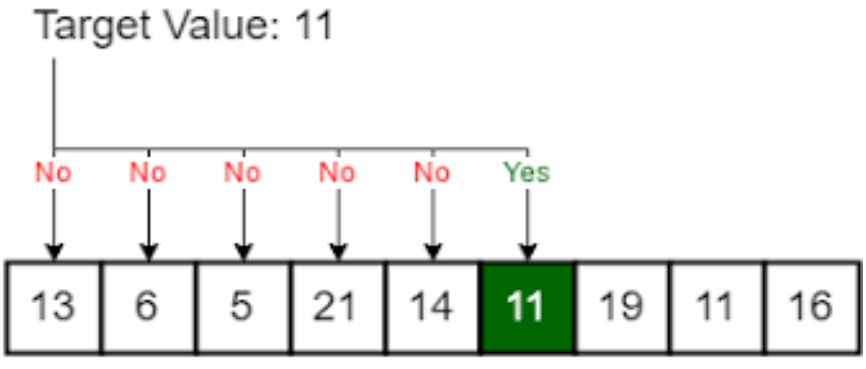


Figure: Linear Search

Sentinel search: It is similar to linear search, is **a sequential search algorithm**. Sentinel Linear Search as the name suggests is a type of Linear Search where the number of comparisons is reduced as compared to a traditional linear search. When a linear search is performed on an array of size N then in the worst case a total of N comparisons are made when the element to be searched is compared to all the elements of the array and (N + 1) comparisons are made for the index of the element to be compared so that the index is not out of bounds of the array which can be reduced in a Sentinel Linear Search.

In this search, the last element of the array is replaced with the element to be searched and then the linear search is performed on the array without checking whether the current index is inside the index range of the array or not because the element to be searched will definitely be found inside the array even if it was not present in the original array since the last element got replaced with it. So, the index to be checked will never be out of bounds of the array. The number of comparisons in the worst case here will be (N + 2).

Examples:

Input: arr[] = {10, 20, 180, 30, 60, 50, 110, 100, 70}, x = 180

Output: 180 is present at index 2

Input: arr[] = {10, 20, 180, 30, 60, 50, 110, 100, 70}, x = 90

Output: Not found

Conclusion: Linear Search and Sentential search are the searching methods that can be efficiently use for searching the elements in array but moth methods required large amount of time to search the elements as they compare key elements with every element in array.

Frequently asked questions:

What is searching?

What is linear search?

What is time complexity of linear search?

Practical No: 4 (B)

Title: Write a python program to search roll numbers of student in array.

Problem Statement: Write a Python program to store roll numbers of student array who attended training program in sorted order. Write function for searching whether particular student attended training program or not, using Binary search and Fibonacci search.

Objective:

- Understand the concept of linear data structure "ARRAY" and how to declare & Use it.
- Understand the basic concepts of searching, types of searching and implementations of searching.

Outcomes:

- Use algorithms on various linear data structure using sequential organization to solve real life problems.
- Understood how to implement Linear Search.
- Understood how to implement Sentinel search.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Theory:

- **Divide and Conquer:** Is a algorithm design strategy in which we can break a single big problem into smaller sub-problems, and solve the smaller sub-problems and combine their solutions to find the solution for the original big problem, it becomes easier to solve the whole problem. The concept of Divide and Conquer involves three steps:

    1. **Divide** the problem into multiple small problems.
    2. **Conquer** the sub problems by solving them. The idea is to break down the problem into atomic subproblems, where they are actually solved.
    3. **Combine** the solutions of the subproblems to find the solution of the actual problem

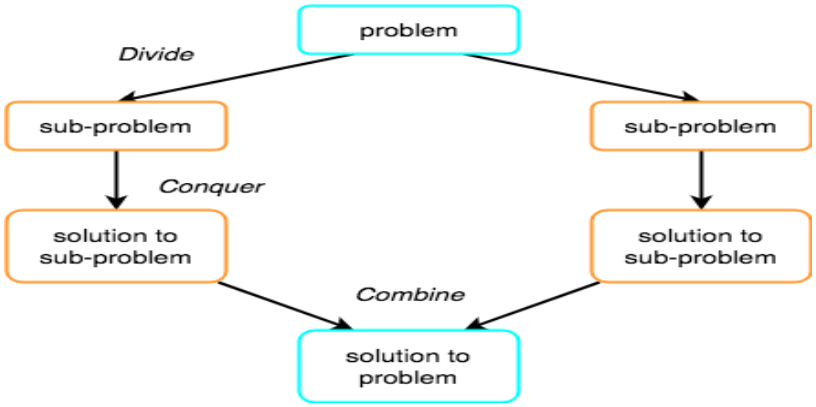


Figure: Concept of Divide and conquer

Binary Search: Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

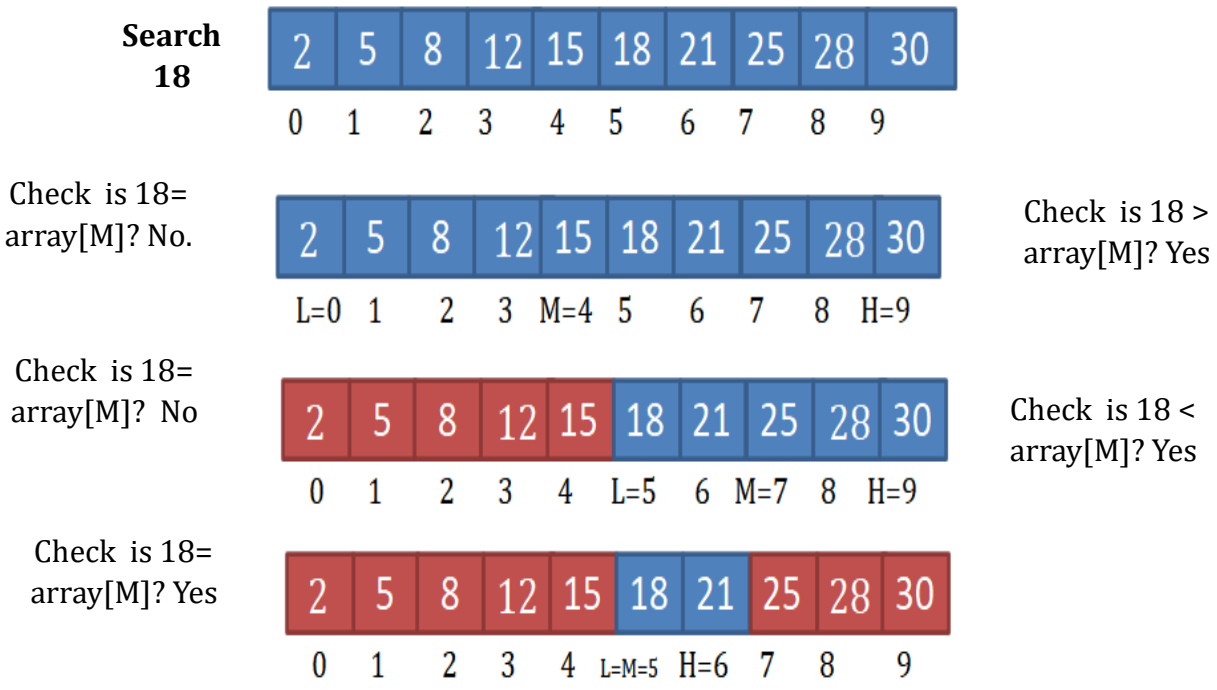


Figure: Binary Search Example

Binary Search Pseudo code :

numbers to Procedure binary_search

```
A ← sorted array
n ← size of array
x ← value to be searched
Set lowerBound = 0
Set upperBound = n-1
while x not found
        if upperBound < lowerBound
            EXIT: x does not exists.
        set midPoint = (lowerBound + upperBound)  / 2
        If A[midPoint] < x
        set lowerBound = midPoint + 1

if A[midPoint] > x
            set upperBound = midPoint - 1
        if A[midPoint] = x
            EXIT: x found at location midPoint
end while
 end procedure
```

Fibonacci Search.: Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array. Time Complexity is O(logn).

Differences with Binary Search:

- Fibonacci Search divides given array in unequal parts
- Fibonacci Search doesn't use /, but uses + and -. The division operator may be costly on some CPUs.
- Fibonacci Search examines relatively closer elements in subsequent steps. So when input array is big that cannot fit in CPU cache or even in RAM, Fibonacci Search can be useful.

- Steps to Search element using Fibonacci search.
- Compute three variables a, b, f Here Initially f=n.
- Construct Fibonacci series to determine a & b.

Fibonacci Series FB=x1, x2, x3, x4, x5……..x5

So a=x1 & b= x2

With this initial values start searching the element, and check the follo wing two conditions

1. if(Key<arr[f])	2. if(Key>arr[f])
f=f-a;	f=f+a;
b=a;	b=b-a;
a=b-a;	a=a-b;

Conclusion: Binary Search and Fibonacci search are the searching methods that can be efficiently use for searching the elements in array and both methods required less amount of time to search the elements as compared to sequential search.

Frequently asked questions:

1. What is searching?
2. What is Binary Search?
3. What is Fibonacci Search?
4. What is time complexity of Binary search?

Practical no-05

Title:

Problem Statement: Write a pythonprogram to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using

a) Selection Sort

b) Bubble sort and display top five scores

Objective:

- Understand the concept of Selection sort using array data structure.
- Understand the concept of Bubble sort using array data structure.

Outcomes:

- Will be able to sort the elements in array using Selection sort.
- Will be able to sort the elements in array using Bubble sort.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

Thoery:

Sorting:

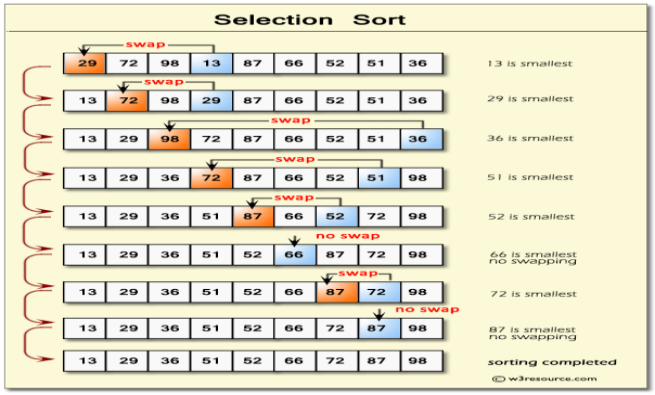
Sorting is systematic arrangement of data.
We are introduces two sorting algorithm in this assignment.

1. Selection Sort
2. Bubble Sort

Selection Sort

Selection sort is a simple sorting algorithm. This sorting algorithm is a in-place comparison based algorithm in which the list is divided into two parts, sorted part at left end and unsorted part at right end. Initially sorted part is empty and unsorted part is entire list.
Smallest element is selected from the unsorted array and swapped with the leftmost element and that element becomes part of sorted array. This process continues moving unsorted array boundary by one element to the right.
This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n2) where n are no. of items.

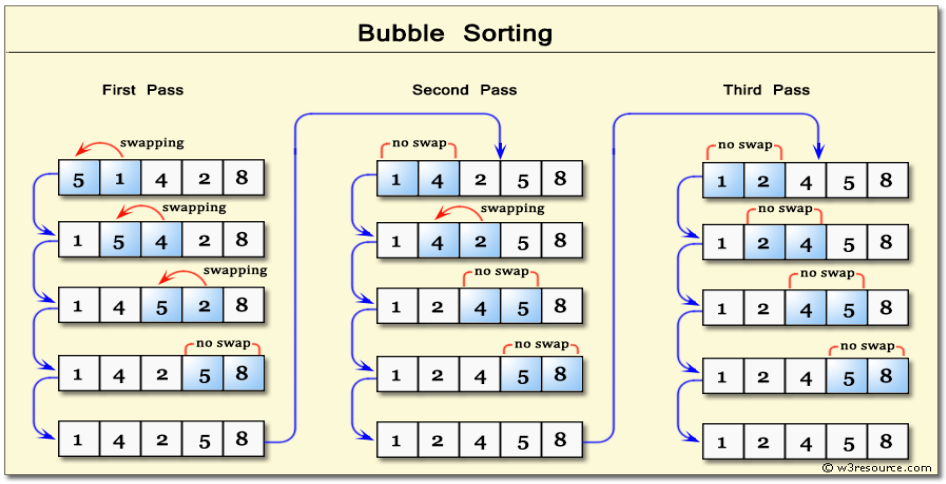
Following is a pictorial depiction of the entire sorting process −



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 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(n2) where n are no. of items.

Following is a pictorial depiction of the entire sorting process −



Algorithm:

Selection Sort:

1. Set MIN to location 0.
2. Search the minimum element in the list.
3. Swap with value at location MIN.
4. Increment MIN to point to next element.
5. Repeat until list is sorted.

Bubble Sort:

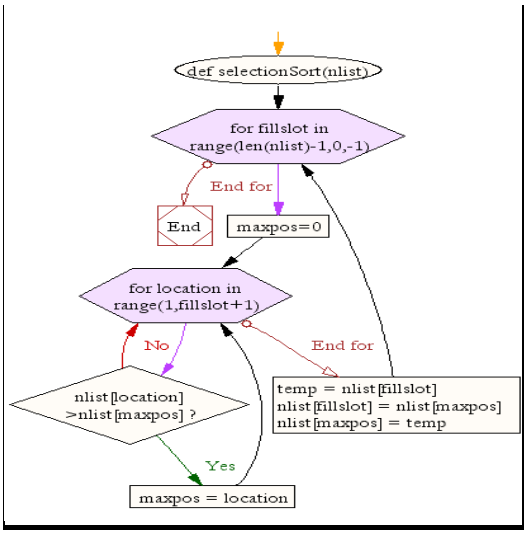
1. Start with all the elements n.

2. Set i=0.
3. Compare the adjacent elements.
4. Repeat step 3 for all the n elements.
5. Increment the value of ‘i’ by 1 and repeat step 3,4 for i<n.
6. Print the sorted list of elements.
7. Stop.

Pseudo code for Selection Sort:

```
defselectionSort(nlist):
forfillslot in range(len(nlist)-1,0,-1):
maxpos=0
for location in range(1,fillslot+1):
ifnlist[location]>nlist[maxpos]:
maxpos = location
temp = nlist[fillslot]
nlist[fillslot] = nlist[maxpos]
nlist[maxpos] = temp
nlist = [14,46,43,27,57,41,45,21,70]
selectionSort(nlist)
print(nlist)
```

Flowchart For Selection Sort:

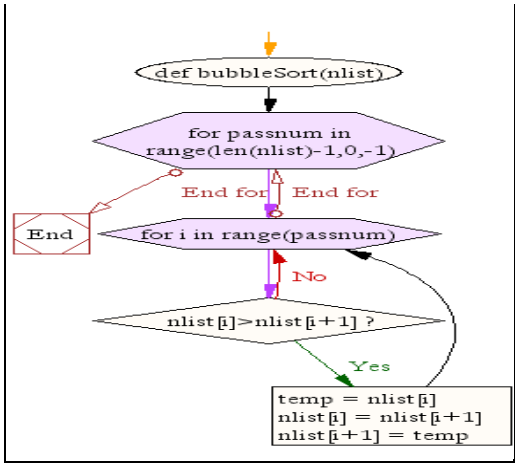


Pseudo code for Bubble Sort:

```
defbubbleSort(nlist):
forpassnum in range(len(nlist)-1,0,-1):
for i in range(passnum):
ifnlist[i]>nlist[i+1]:
temp = nlist[i]
nlist[i] = nlist[i+1]
nlist[i+1] = temp
```

```
nlist = [14,46,43,27,57,41,45,21,70]
bubbleSort(nlist)
print(nlist)
```

Flowchart for Bubble Sort:



Conclusion:

Hence from the above theory we understood the concept of Selection and Bubble sort and can be able to implement using array data structure and display top five elements.

Viva Questions:

1. Discuss time complexity of bubble sort
2. Discuss time complexity of selection sort

Practical No: 6

Title: Write a python program to sort the given data.

Problem Statement: Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.

Objective:

- Understand the concept of linear data structure and how to declare & use it.
- Understand the basic concepts of Sorting, types of Sorting and implementations of Sorting .
- Understand the basic concepts of Quick Sort and implementations of it .

Outcomes:

- Use algorithms on various linear data structure using sequential organization to solve real life problems.
- Analyze problems to apply suitable searching and sorting algorithm to various applications Understood how to implement Quick sort.

Software & Hardware Requirement:

- 64-bit Open source Linux or its derivative
- Open Source python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

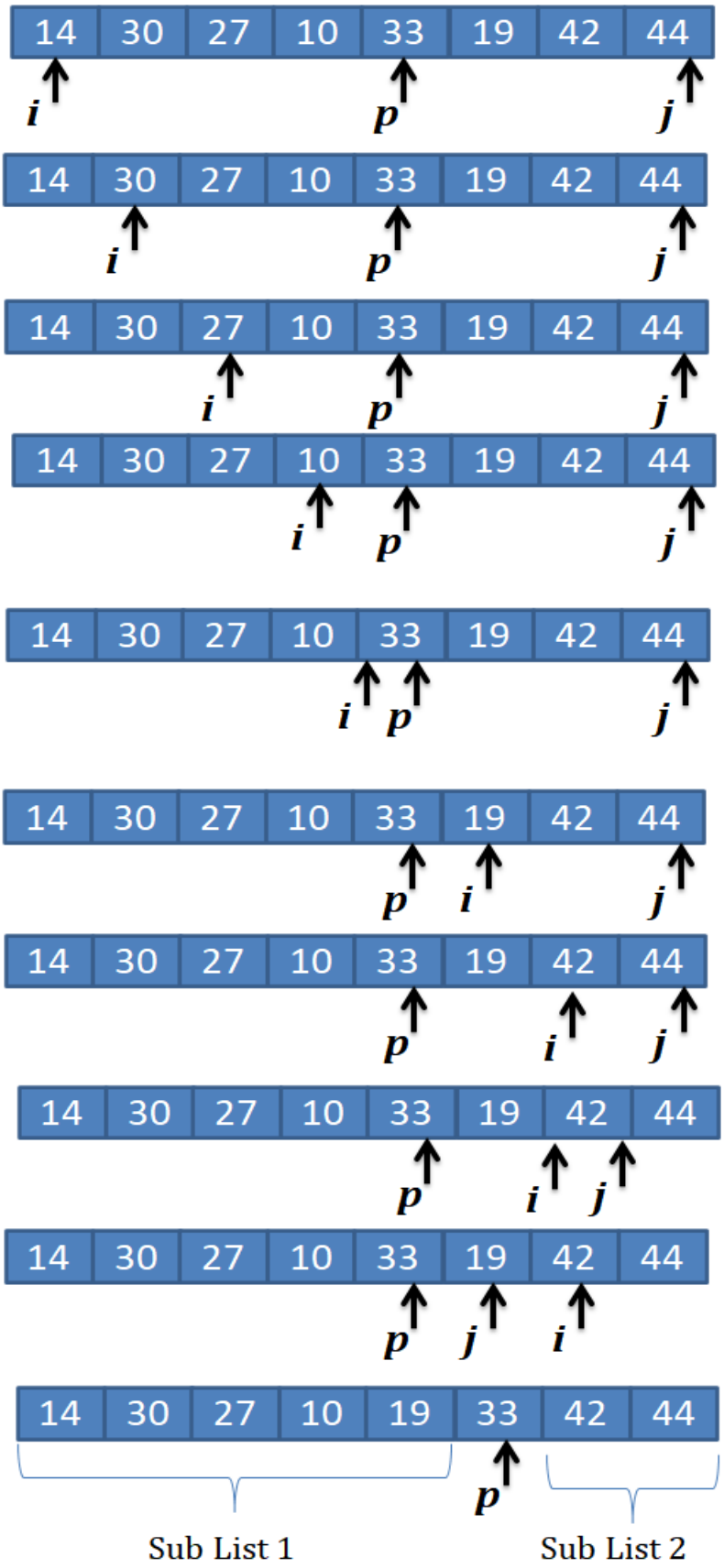
Theory:

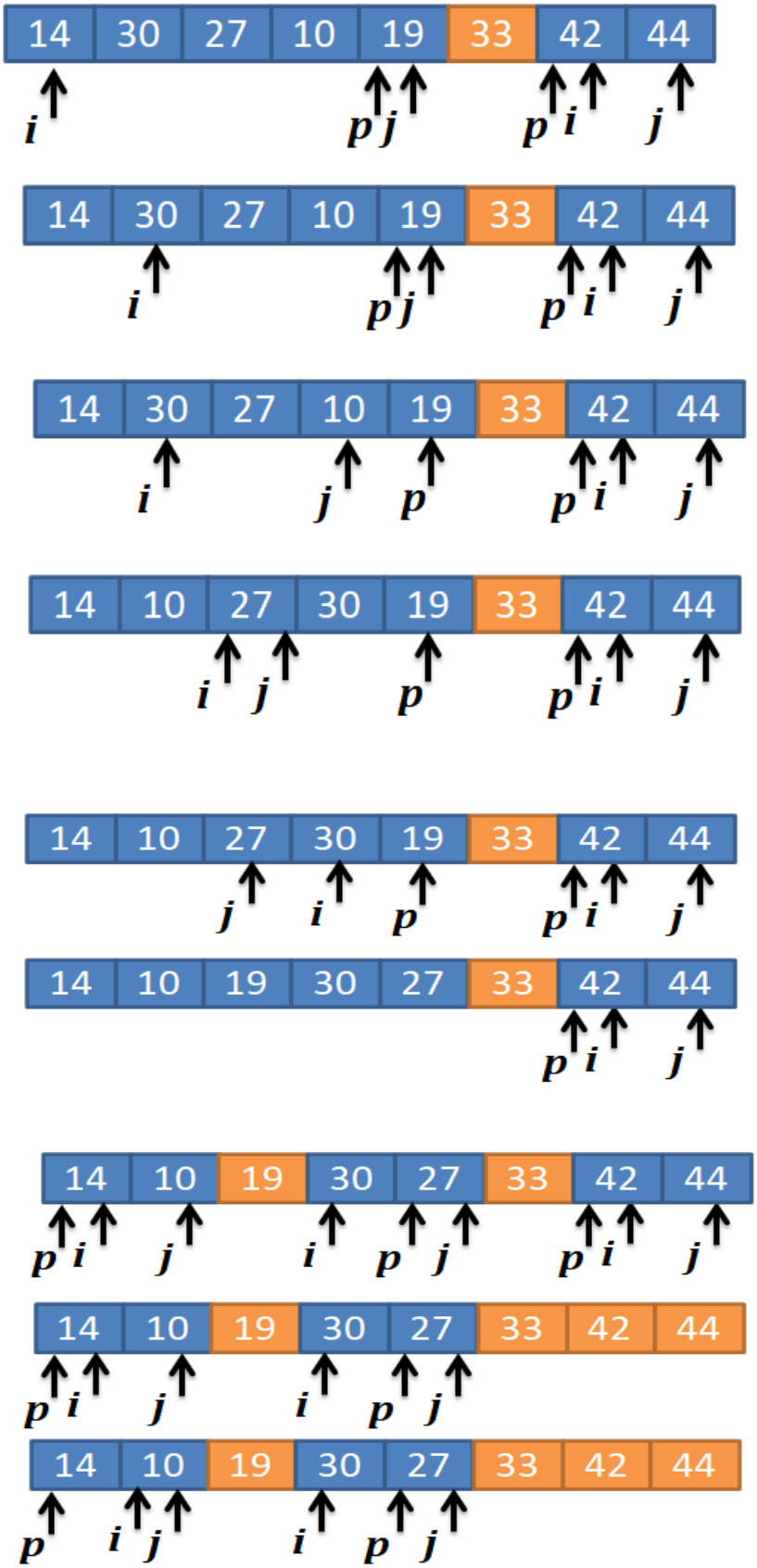
- **Searching** is the process of finding a given value position in a list of values. It decides whether a search key is present in the data or not. It is the algorithmic process of finding a particular item in a collection of items.
- Different Searching techniques are,
    1. Linear Search.
    2. Binary Search.
    3. Fibonacci Search.

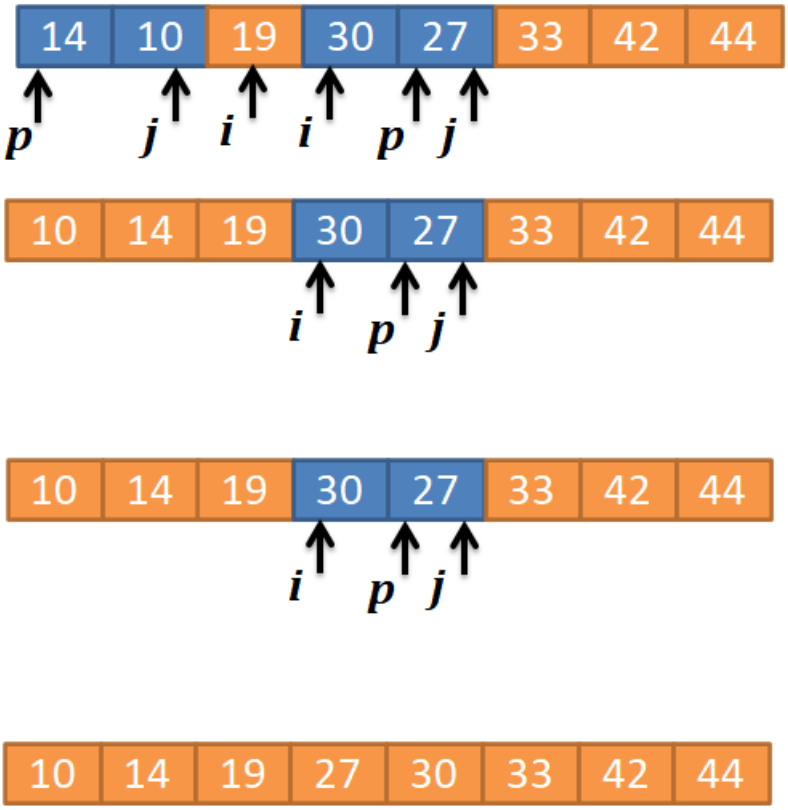
Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value. It is implemented using divide & conquer strategy.

The time complexity for Quick sort is O(nlogn) for best case and for worst case it is $O(n^2)$.

Ex. Let us Sort the given elements 14, 30, 27, 10, 33, 19, 44, 42 Using Quick Sort.


14 30 27 10 33 19 42 44
i p j
14 30 27 10 33 19 42 44
i p j
14 30 27 10 33 19 42 44
i p j
14 30 27 10 33 19 42 44
i p j
14 30 27 10 33 19 42 44
i p j
14 30 27 10 33 19 42 44
p i j
14 30 27 10 33 19 42 44
p i j
14 30 27 10 33 19 42 44
p i j
14 30 27 10 33 19 42 44
p j i
14 30 27 10 19 33 42 44
p
Sub List 1
Sub List 2


14 30 27 10 19 33 42 44
i p j p i j
14 30 27 10 19 33 42 44
i p j p i j
14 30 27 10 19 33 42 44
i j p p i j
14 10 27 30 19 33 42 44
i j p p i j
14 10 27 30 19 33 42 44
j i p p i j
14 10 19 30 27 33 42 44
p i j i p j p i j
14 10 19 30 27 33 42 44
p i j i p j
14 10 19 30 27 33 42 44
p i j i p j



Quick Sort Algorithm:

1. **Start**
2. **Declare all the required variables and array of size n.**
3. **Read the elements in array.**
4. **After selecting an element as pivot, which is the last index of the array in our case, we divide the array for the first time.**
5. **In quick sort, we call this partitioning. It is not simple breaking down of array into 2 subarrays, but in case of partitioning, the array elements are so positioned that all the elements smaller than the pivot will be on the left side of the pivot and all the elements greater than the pivot will be on the right side of it.**
6. **And the pivot element will be at its final sorted position.**
7. **The elements to the left and right, may not be sorted.**
8. **Then we pick subarrays, elements on the left of pivot and elements on the right of pivot, and we perform partitioning on them by choosing a pivot in the subarrays.**
9. **Print the sorted array list**
10. **Stop**

Quick Sort Pseudo code:

```
partition (arr[], low, high)
{
        pivot = arr[high];
        i = (low - 1) // Index of smaller element
        for (j = low; j <= high- 1; j++)
        { // If current element is smaller than the pivot
                if (arr[j] < pivot)
                { i++; // increment index of smaller element
                swap arr[i] and arr[j]
        }
        } swap arr[i + 1] and arr[high])
return (i + 1)
}

quickSort(arr[], low, high)
{
        if (low < high)
        { /* pi is partitioning index,  arr[pi] is now at right place */
                pi = partition(arr, low, high);
                quickSort(arr, low, pi - 1); // Before pi
                quickSort(arr, pi + 1, high); // After pi
        }
}
```

Conclusion:

Quick sort is most efficient method of sorting with respect to other sorting strategies and time complexity point of view.

Frequently asked Questions:

What is quick Sort?

What is time complexity of quick sort?

Practical No -07

Title: Write a C++ program to create pinnacle club data base using Singly Linked List.

Problem Statement: Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to:

a) Add and delete the members as well as president or even secretary.
b) Compute total number of members of club
c) Display members
d) Two linked lists exists for two divisions. Concatenate two lists.

Objectives::

- To study the dynamic memory allocation.
- To understand Singly Linked List.
- To study theconcept of concatenation of two singly linked lists.

Outcome:

- Will be able to use conceptof singly linked list in programming.
- Will be able to understand concept of Sequential Data Structure**.**

Software & Hardware Requirements:

- 64-bit Open source Linux or its derivative
- Open Source C++ Programming tool like G++/GCC.

Theory:

Dynamic Memory Allocation ::

In case of dynamic data structures, the memory space required variables is calculated and allocated during execution. Dynamic memory in managed in ‘C++’ through a set of library function.

- ✓ **Allocating a block of memory in “C++”**

ptr =(cast-type)new(byte-size); The “new ()” returns a pointer (of cast type) to an area ofmemory with size, byte-size

- ✓ **Example::**x=(int*)new(100*sizeof(int));

2)Singly Linked List ::

In this type of linked list two successive nodes of the linked list are linked with each otherin sequential linear manner. Movement in forward direction is possible

a) Create Function

Assume n=3 (3 nodes to be created) with inputs as 5,1,9. The address of the newlyacquired node pointed by P. Subsequently, P is moved to the next node.

E.g.

```
#include <iostream.h>
typedefstruct node
```

```
{
        int data; struct node *next;
}node; node*
createnode ()
{
        node*head,*p,*q;
        inti,no; head=NULL;
        cout<<"\nEnter the no ofnodes::"; cin>>"%d",&no;
                for(i=0;i <no;i++)
                {
                        p=(node*)malloc(sizeof(node));
                        cout<<"\nEnter the data::";
                        cin>>"%d",&(p->data; p->next=NULL;
                        if(head==NULL)
                                head=p;
                        else
                        {
                                q=head;
                                while(q->next!=NULL )
                                        q=q->next;
                                        q->next=p; } } return head;
                        }
```

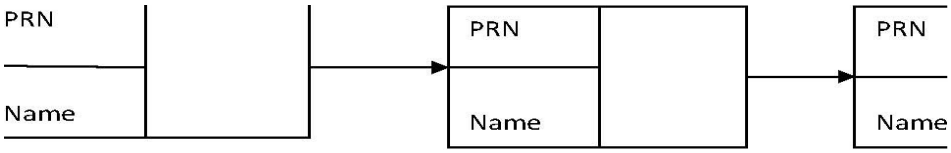
b)Print Function

```
        void print (node*head)
        {

                node*p;

                int count=0;
                cout<<"\nList of elements are ::";
                p=head;
                while(p!=NULL)
                {
                        count++;
                        cout<<"%d=>",p->data;
                        p=p->next;
                }
```

```
        Cout<<"\nTotal no of elements=%d",count;
}
```



b) Insert at beginning

Algorithm

1. Obtain space for new node.
2. Assign data to the data field of the new node.
3. Set the next field ofthe new node to the beginning of the linked list.
4. Change the reference pointer of the linked list to pointto the new node.

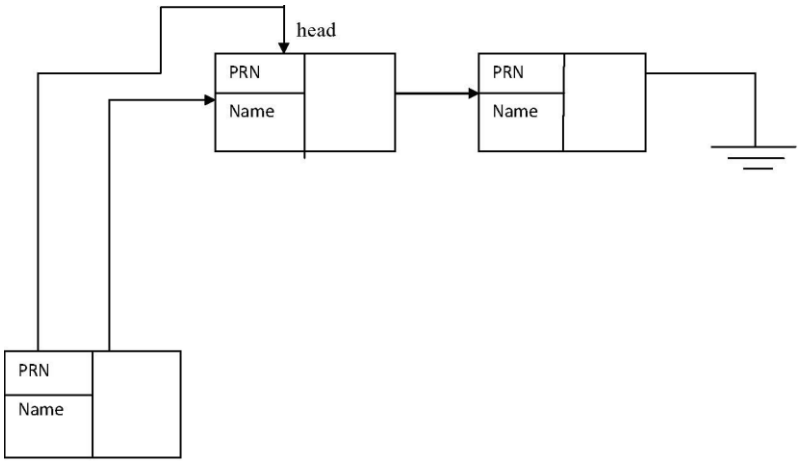
E.g.

```
node*insertb (node*head,intele)
{
        node *p;
        p=(node*)malloc(sizeof (node));
        p->data=ele; p->next=NULL;
        if(head==NULL)
        {
                head=p;
                return head;
                }
                else
                {
                        p->next=head;

                        head=p;

                        return head;

                }
```



d)Insert at location

Algorithm

1.Acquire memory for new node with its address in pointer P.

i.e.P = (node *) malloc(sizeof(node));

2.Assign value to data field and make its ‘next’ field ‘NULL’.

i.e.P data =x; P next = NULL;

3.if (LOC == 1)

Theninsert the node, pointed by P, at thebeginning of the linked list. This can be done byfollowing steps.

(a)Store head in the next field of the node pointed by P

1 P ->next = head;

(b)Move head to the newly connected node.

2 head = P; go to step 7 4.[If Loc> 1] Position a pointer q on (LOC -1)th node .

3 q = head; for (i = 1; i < (Loc -1); i++) q = q-> next;

5. if q is NULL then report “overflow” andterminate the algorithm.Go to step 7
6. Insert the node pointed by P, after the node pointed by q.

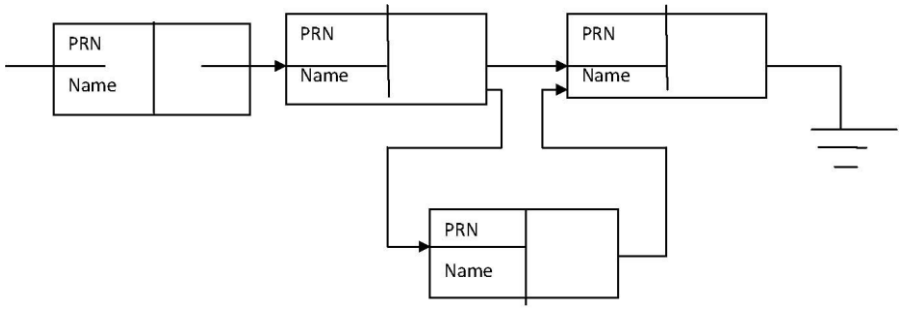
4 P ->next = q->next; q ->next = p;

7.Stop.

E.g.

```
node *insertl(node*head,intele,intloc)
{
    node *p,*q; int i; p=(node*)malloc(sizeof(node)); p->data=ele;
    p->next=NULL; if (loc==1)
    {
    p->next=head; head=p;
```

```
        return head;
        }
        q=head; for (i=1;i<loc-1;i++)
if (q->next!=NULL)
        q=q->next;
else
{
        cout<< "\nOverflow"; return
        head;
}
p->next=q->next;
q->next=p;
return head;
}
```



e)Insert at end Algorithm

Insert an item 'x' in a linked list, referenced by the pointer 'head'

1.Acquire memory for new node

i.e.p = (node*) malloc(sizeof (node);

2.Assign value to the data filed and make its 'next' field 'NULL'

i.e.P□ data = x; P □ next = NULL;

3.If 'head' is 'NULL' Thenhead = p ; goto step 6

4.Positionapointer q on the last node by traversing the linked list form the first node anduntil it reaches the last node.i.e. q= head while (q □ next ! = NULL) q= q □ next ;

5.Store the address of the newly acquired node,pointed by P, in the next field of node pointed by q.

i.e. q □ next = p;

6.Stop.

E.g.

```
node*inserte(node*head,intele)
{
        node *p,*q; p=(node*)malloc(sizeof (node));
        p->data=ele;
        p->next=NULL;
```

```
if(head==NULL)
{
        head=p; return head;
}
for(q=head;q->next!=NULL; q=q->next);
        q->next =p;
return head;
}
```

f) Delete at beginning Algorithm

1.Store the address of the first node in a pointer variable, say P.

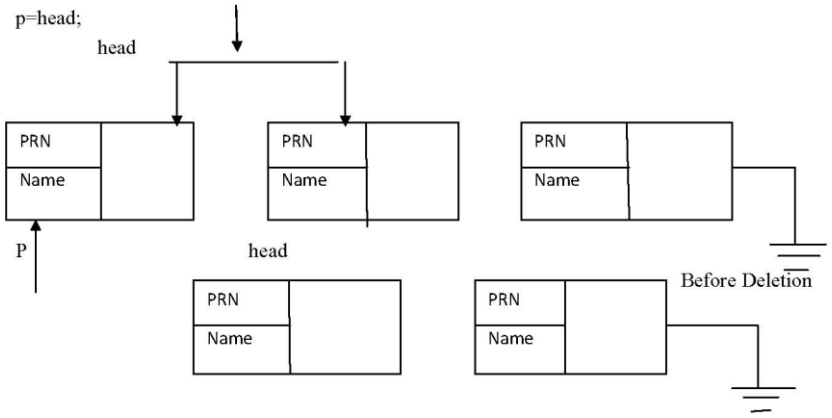
2.Move the head to the next node.

3.Freethe node whose address is stored in the pointer variable P.

E.g.

```
node *delb(node*head)
{ node *p;
        if(head==NULL)
        {
                cout<< "\nList is empty";
                return head;
        }
After Deletion
head=head->next;

p->next=NULL;
delete p;
return head; }
```



g)Delete at location

Algorithm

1.Store the address of the preceding nodein a pointer variable P. Node to be deleted is marked as key node.

2.Store the address of the key node in a pointer variable q, so that it can be freed

subsequently.

3.Make the successor of the key node as the successor of the node pointed by P.

4.Freethenode whose address is stored in the pointer variable q.

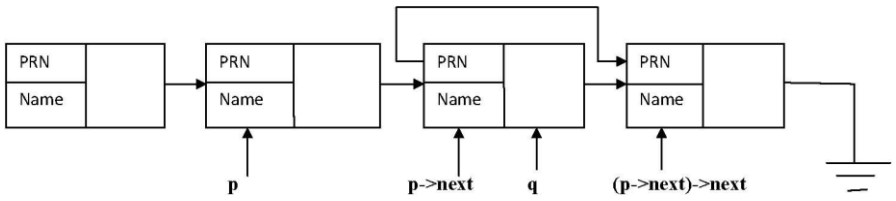
E.g.

```
node *dell(node *head,intloc)
{
        node*p,*q;
        int i;
                if (loc==1)
                {
                        p=head;
                        head=head->next;
                        p->next=NULL;
                        delete p;
                        return head;
                }
                q=head;
for (i=1;i<loc-1;i++);
if(q!=NULL)
        q=q->next;
else
{
        Cout<< "\nUnderflow"; return head;
}
p=q->next;
p->next=p->next;
p->next=NULL;

delete p;

return head;

}
```



H)Delete at end Algorithm

[Deleting last node of a linked list, referenced by the pointer head]

In order to delete the last node, we must position a pointer q on the last but one node. Address of the nodeto be deleted is stored in pointer P, So that the memory allocated to it can be freed.

1. If the firstnode itself is the last node then [make the linked list empty]

1 if (head □ next == NULL)

{

free (head); head = NULL;

goto

step 4 } 2.[otherwise] position a

pointer q on last but one node

2 q = head; while (q->next ->next! = NULL) q = q-> next; 3.Delete the last node

3 p = q->next; free (p);

q ->next = NULL; 4.Stop.

E.g.

```
Node*dele(node*head)
{ node*p,*q;
if(head==NULL)
{
        cout<< "\nList is empty"; return head;
}
for (q=head;q->next->next!=NULL;q=q->next); p=q->next; q->next=NULL; delete p;
return head; }
```

i)Concatenate two lists

Algorithm

Let us assume that the two linked listsare referenced by head1 and head2 respectively.

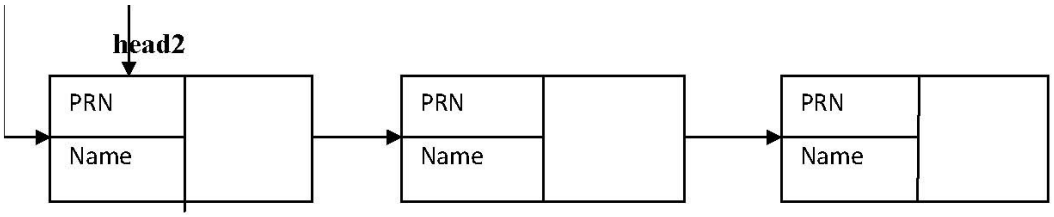
(1)If the firstlinked list is empty then return head2.

(2)If the second linked list is empty then return head1.

(3)Store the address of the starting node of the first linked list in a pointer variable, say P.

(4)Move the P to the last node of the linked list through simple linked list traversal technique.

(5)Store the address of the first node of the second linked list in the next field of the node pointedby P. Return head1



E.g.

node * concatenate(node *head,node *head2) { node *p; if (head1==NULL) return head 2;

if (head2==NULL)

return head 1;

p=head 1;

while(p->next!=NULL)

Conclusion:

By using Singly Linked List and concept of concatenation we have successfully implemented the program.

Viva Questions:

1. Define link list?
2. Explain different types of link list
3. Difference between Singly link list and Doubly Link list

Practical No -08

Title: Write a C++ program to create pinnacle club data base using Doubly Linked List.

Problem Statement: The ticket booking system of Cinemax theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand
a) The list of available seats is to be displayed
b) The seats are to be booked
c) The booking can be cancelled.

Objectives:

- To study the dynamic memory allocation.
- To understand Singly Linked List.
- To study the concept of concatenation of two singly linked lists.

Outcome:

- Will be able to use concept of doubly linked list in programming.
- Will be able to understand concept of Sequential Data Structure**.**

Software & Hardware Requirements:

- 64-bit Open source Linux or its derivative
- Open Source C++ Programming tool like G++/GCC.

Theory:

Dynamic Memory Allocation ::

In case of dynamic data structures, the memory space required variables is calculated and allocated during execution. Dynamic memory in managed in 'C++' through a set of library function.

- ✓ **Allocating a block of memory in "C++"**

ptr =(cast-type)new(byte-size); The "new ()" returns a pointer (of cast type) to an area ofmemory with size, byte-size

- ✓ **Example::**x=(int*)new(100*sizeof(int));

2)Singly Linked List ::

In this type of linked list two successive nodes of the linked list are linked with each otherin sequential linear manner. Movement in forward direction is possible

a) Create Function

Assume n=3 (3 nodes to be created) with inputs as 5,1,9. The address of the newlyacquired node pointed by P. Subsequently, P is moved to the next node.

E.g.

```
#include <iostream.h>
typedef struct node
{
```

```
        int data; struct node *next, *prev;
}node; node*
void cinemax::create_list()
{
        int i=1;
        temp=new node;
        temp->seat=1;
        temp->status=0;
        temp->id="null";
        tail=head=temp;
        for(int i=2;i<=70;i++)
        {
                node *p;
                p= new node;
                p->seat=i;
                p->status=0;
                p->id="null";
                tail->next=p;
                p->prev=tail;
                tail=p;
                tail->next=head;
                head->prev=tail;

        }
}
```

b) Print Function

```
        oid cinemax::display()
        {
        {
                 int r=1;
                node* temp;
                temp=head;
                int count=0;
                cout<<"\n--------------------------------------------------------------------------------\n";
                cout<<" Screen this way \n";
                cout<<"--------------------------------------------------------------------------------\n";
                while(temp->next!=head)
```

```
        {
                if(temp->seat/10==0)
                cout<<"S0"<<temp->seat<<" :";
                else
                cout<<"S"<<temp->seat<<" :";

                if(temp->status==0)
                cout<<"|___| ";
                else
                cout<<"|_B_| ";
                count++;
                if(count%7==0)
                {
                cout<<endl;
                r++;
                }
                temp=temp->next;
        }
        cout<<"S"<<temp->seat<<" :";
        if(temp->status==0)
        cout<<"|___| ";
        else
        cout<<"|_B_| ";
        }
}
```

c) Book Ticket

Algorithm

1. Ask User to enter seat number to be booked.
2. Check weather seat no is a valid seat number Between 1 to 70
3. Check current status of seat, if already booked.
4. Ask to enter Valid ID Number to book seat
5. Book seat if seat available.

E.g.

```
void cinemax::book()

{
```

```
int x;
string y;
label:
cout<<"\n\n\nEnter seat number to be booked\n";
cin>>x;
cout<<"Enter your ID number\n";
cin>>y;
if(x<1||x>70)
                {
                cout<<"Enter correct seat number to book (1-70)\n";
                goto label;
                }
                node *temp;
                temp=new node;
                temp=head;
                while(temp->seat!=x)
                {
                        temp=temp->next;
                }

                if(temp->status==1)
                cout<<"Seat already booked!\n";
                else{
                temp->status=1;
```

```
temp->id=y;
cout<<"Seat "<<x<<" booked!\n";
}

}
```

d) Cancel Booking

1. Ask User to enter seat number to be cancelled.
2. Check weather seat no is a valid seat number Between 1 to 70
3. Check current status of seat,
4. Ask to enter Valid ID Number to book seat
5. Cancel the seat booing if seat status is booked.

E.g.

```
void cinemax::cancel()
{
    int x;
    string y;
    label1:
    cout<<"Enter seat number to cancel booking\n";
    cin>>x;
    cout<<"Enter you ID\n";
    cin>>y;
    if(x<1||x>70)
    {
    cout<<"Enter correct seat number to cancel (1-70)\n";
    goto label1;
    }
    node *temp;
    temp=new node;
    temp=head;
    while(temp->seat!=x)
    {
    temp=temp->next;
```

```
        }
        if(temp->status==0)
        {
        cout<<"Seat not booked yet!!\n";
        }
        else
        {
        if(temp->id==y)
        {
        temp->status=0;
        cout<<"Seat Cancelled!\n";
        }

        else
        cout<<"Wrong User ID !!! Seat cannot be cancelled!!!\n";
        }
}
```

Conclusion:

Doubly Linked List can be used to implement various real time problems. In this problem we have implemented it f or the ticket booking system of Cinemax theatre.

Viva Questions:

1. Define Doubly linked list?
2. Explain different types of memory management.
3. Difference between Singly link list and Doubly Link list

Assignment No.- 9

Title:-

Write a C++ program using stack to perform various operations.

Objectives:-

1) To check the given string is palindrome or not by using stack.

Problem Statement:-

A palindrome is a string of character that‘s the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, “Poor Dan is in a droop” is a palindrome, as can be seen by examining the characters “poor danisina droop” and observing that they are the same forward and backward. One way to check for a palindrome is to reverse the characters in the string and then compare with them the original-in a palindrome, the sequence will be identical. Write C++ program with functions

a)To print original string followed by reversed string using stack

b) To check whether given string is palindrome or not.

Software Requirements:-

Ubuntu Linux 14.04,GCC / G++ (Editor).

Theory and Concepts:

1. **Strings**

String is a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null. The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

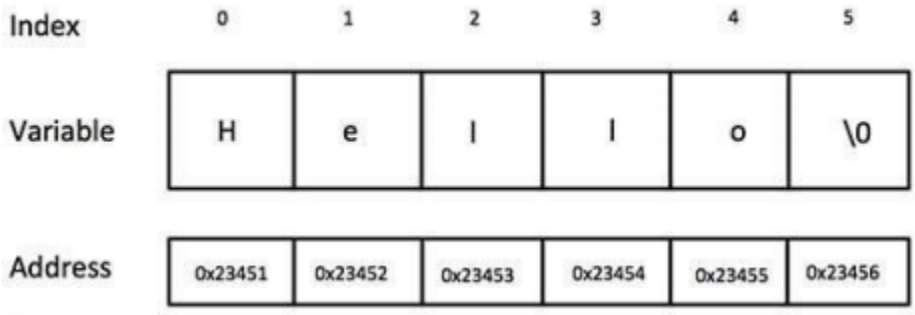
If you follow the rule of array initialization, then you can write the above statement as follows –

```
char greeting[] = "Hello";
```

Input:-

String of characters

Following is the memory presentation of above defined string in C/C++ – String Presentation in C/C++



Palindrome:

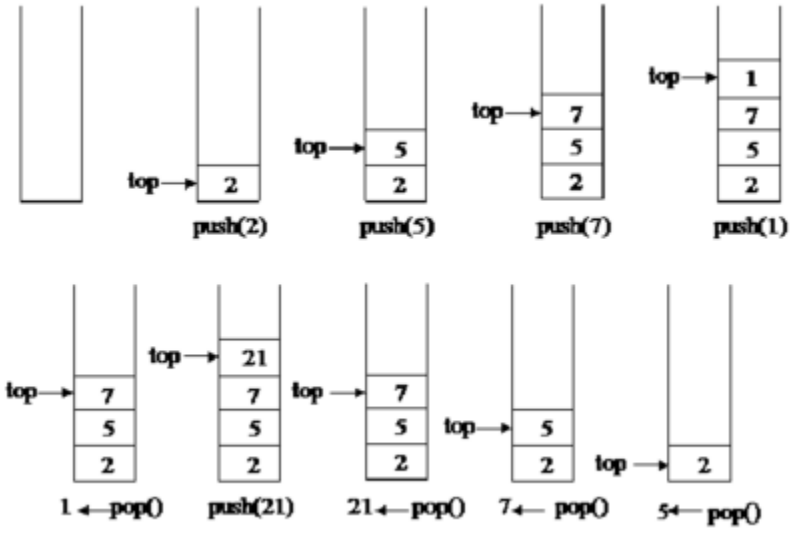
A string is said to be palindrome if reverse of the string is same as string. For example, "abba" is palindrome, but "abbc" is not palindrome.

Stacks

Stack is a LIFO (Last In First Out) data structure. It is an ordered list of same type of elements. A stack is a linear data structure where all insertions and deletions are permitted only at one end of the list. When elements are added to stack it grows at one end. Similarly, when elements are deleted from a stack, it shrinks at the same end.

Stack As an ADT

Stack is a LIFO structure. Stack can be represented using an array. A one dimensional array can be used to hold elements of a stack. Another variable "top" is used to keep track on the index of the top element.



Formally, a stack may be defined as follows:

```
typedef struct stack
{
 int data[MAX];
 int top;
};
```

Input:

String of characters.

Output:-
Display the Reverse the string

Check string is palindrome or not

Algorithm:
Student has to write algorithm.

Flowchart:

Student has to draw the flowchart.

- **Conclusion:**

Thus, we have implemented C++ program for stack operations.

- **Viva Questions:**
    1. What is stack?
    2. What are the operations on stack?
    3. Which are the applications of stack?
    4. Explain expression conversion.
    5. What is recursion in a stack?

Assignment No.- 10

Title:-

Write a C++ program using stack whether given expression is well parenthesized or not.

Objectives:-

1) To check the given expression is parenthesized or not by using stack

Problem Statement:-

In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given expression is well parenthesized or not.

Software Requirements:-

Ubuntu Linux 14.04,GCC / G++ (Editor).

Theory and Concepts:

Stack:

A) Stack is a LIFO (last in first out) structure. It is an ordered list of the same type of elements.
B) A stack is a linear list where all insertions and deletions are permitted at only one end of list.
C) When elements are added to stack it grows at one end. Similarly, when elements are deleted from stack, it shrinks at the same end.

1. **Stack Using Array:**
    1. Stack is a LIFO structure. Stack can be represented using array.
    2. A one dimensional array can be used to hold elements of stack.
    3. Another variable "top" is used to keep track of the index of the top most elements.
    4. The following operations can be done on the stack by using array:
        a. **Initialise**
        b. **IsEmpty**
        c. **IsFull**
        d. **Push**
        e. **Pop**
        f. **Print**
        g. **Stack Top**

1. **Initialize:** This operation will initialise the Stack and top pointer to initial position.

```
Void initialize(int stack[10],int top)
{
          top=-1;
}
```

2. **IsEmpty :** This operation will determine if a stack is empty or not.

```
int isEmpty(int stack[10],int top)
{
   if(top==-1)
     {
       return(1);
     }
         return 0;
}
```

3. IsFull : This operation will determine if a stack is Full or not.

```
int isFull(int stack[10],int top)
{
   if(top==MAX-1)
     {
       return(1);
     }
         return 0;
}
```

4. Push :

```
void push(int item)
{
          if (top !=MAX-1)
          {
                    top++; //top incremented by 1
                    stack[top] = item;  //Insert the element from stack
          }
}
```

5. Pop:

```
int pop( int top)
{
     if (top == -1)
       printf("\nstack is empty");
    else
          {
                    item=stack[top];

                    top--;
          }//end else
   return(item);
 }
```

Parenthesized:-

In the assignment created by us the meaning of parenthesized can be defined as in which statement is completed by using opening and closing brackets.

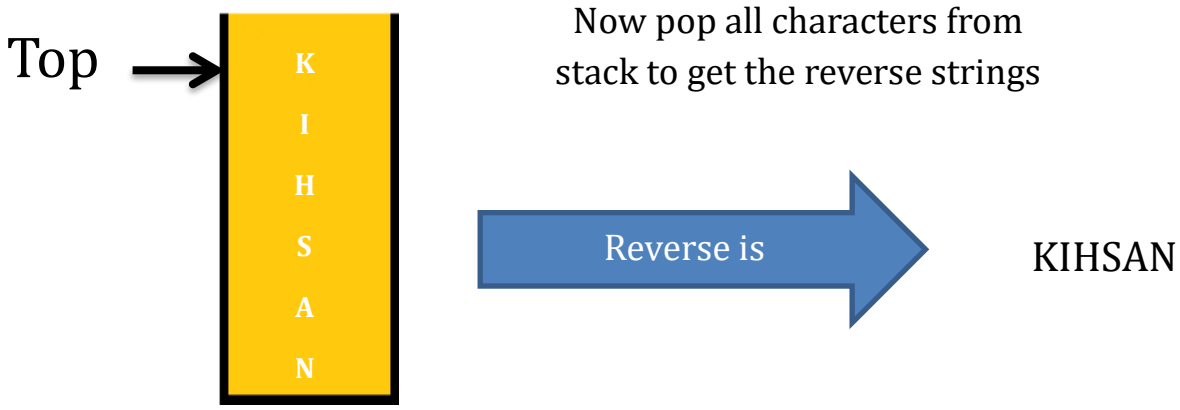
For eg:- 1. (a+b)
2. {a+b}
3. [a+b]
4. {q[h+k(a+v)]}

Following are some of the important applications of a Stack

1. Expression Conversion
2. Expression Evaluation
3. Parsing of parenthesis
4. String Reversal
5. Storing Function Call
6. Decimal to Binary conversion

String Reversal:

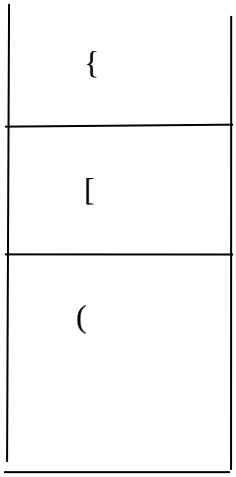
- Stack is used to reverse a string.
- We push the characters of string one by one into stack and then pop character from stack.
    - Example: Let us consider the input string "NASHIK".



Operation:-

We used a stack to complete the operation using parenthesize. When we give a opening bracket in the statement we push the bracket in stack and in the case of closing bracket we pop the top most opening bracket and compare with the closing bracket.

1. If the brackets are equal to each other the case of **valid statement** gets printed in the test case.
2. If the brackets are unequal to each other the case of **invalid statement** gets printed in the test case.
3. If the stack is not empty and a series of closing brackets has been exhausted then also the statement is not well parenthesized.
4. If the stack is empty and series of closing brackets hasn't been exhausted then also the statement is not well parenthesized and **invalid statement** message occurs.



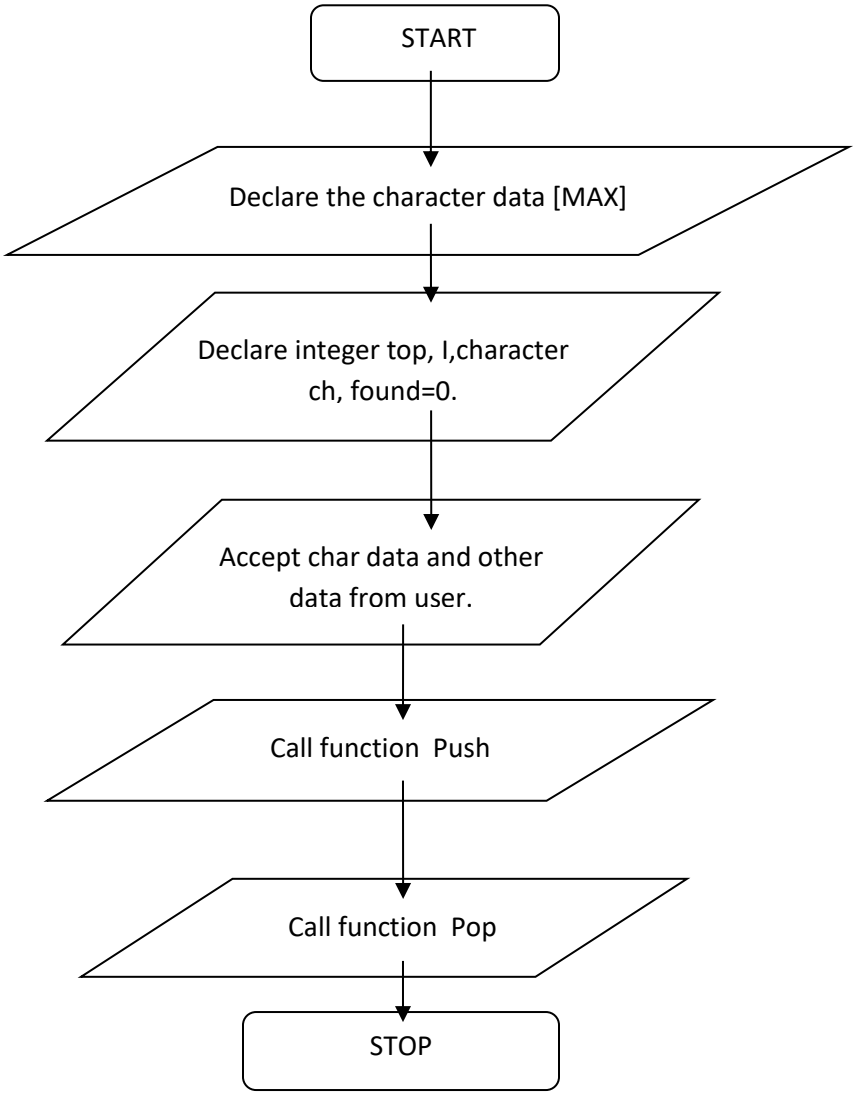
} - closing bracket

In the case given in figure the topmost element in stack is compared with closing bracket. If they match each other then valid statement message Occurs.

Algorithm:-

Step 1:- START.
Step 2:- Declare Character Array Data [MAX].
Step 3:- Declare integer top, I, character ch, found=0.
Step 4:-Accept char data and other data from user.
Step5:- Call function Push
Step6:- Call function Pop
Step7:- STOP

Flowchart:-



Conclusion:-

Hence, We are study and design how to check whether it parenthesized or not and successfully implement the program using stacks.

Viva questions:

a. What is stack?

b. List applications of stack.

Assignment No - 11

Aim: Write a C++ program to simulating job queue by using array and linked list.

Objectives:

1. To study queue using array
2. To study queue using linked list.

Problem Statement:

Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.

Software Requirements:

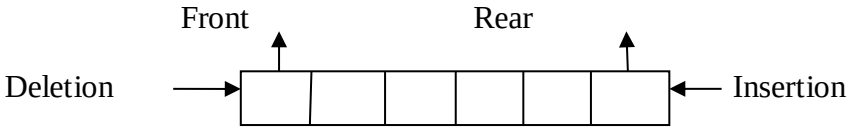
Ubuntu Linux14.04,GCC / G++(editor)

Theory and Concept:

1. **Queue:**

   It is a special kind of list, where items are inserted at one end(the rear) and deleted from the other end(front). Queue is a FIFO(First In First Out) list.

   We come across the term queue in our day to day life. We see a queue at a railway reservation counter, or a movie theatre ticket counter. Before getting the service, one has to wait in the queue. After receiving the service, one leaves the queue. Services is provided at one end (the front) and people join the other end(rear).



2. **Queue Using Array:**

   An array representation of queue requires three entities:

   a. An array to hold queue elements.
   b. A variable to hold the index of the front element.
   c. A variable to hold the index of the rear element.

A queue datatype may be defined formally as follows:

```
# define MAX 30

typedefstruct queue
```

```
        {
int data[MAX];
intfront,rear;
}queue;
```

a. Initialization Of Queue:

```
voidinit(structque *q)
{
        int i;
        for(i=0;i<MAX;i++)
        q->arr[i]=-1;
        q->front=0;
        q->rear=-1;
}
```

b. Print Function:

```
for(i=0;i<MAX;i++)
cout<<"  "<<q.arr[i];
cout<<"/t/tFront="<<q.front<<"    rear="<<q.rear;
cout<<"\n\n";
ch=getchar();
```

c. Isempty Condition:

```
intisempty(que *que.int data)
{
returnq.rear<q.front?1:0;
}
```

d. Isfull Condition:

```
intisfull(structque q)
    {
        Return q.rear==MAX-1?1:0;
    }
```

e. Add Function:

Algorithm for insertion in a queue:

1. Insertion in an empty queue.
   rear=front=0;

data[rear]=x;

2. Inserting in a non-empty queue.
   Rear=rear+1;
   data[rear]=x;

```
void add(structque *q, int data)
{
    q->rear+=1;
    q->arr [q->rear]=data;
}
```

f. Delete Function:

Algorithm for deletion from queue:

1. Deletion of last element or only element(rear=front)
   a) x=data[front]
   b) rear=front=-1;
   c) return x;
2. Deletion of the element when the queue length >1
   a) x=data[front]
   b) front=front+1
   c) return x;

3. Queue Using Linked List:

a. Initialization Function:

```
que::que()
{
front=rear=NULL;
}
```

b. Create Function:

```
cout<<"\nEnter the no. of Jobs you want to create:";

cin>>n;

for(i=0;i<n;i++)

{
```

Cout<<"\nEnter the jobs:";

Cin>>data;

if(front==NULL)

front =rear=new node(data);

else{

rear->next=new node(data);

rear=rear->next;

}

c.Add Function:

Steps required for insertion of element x in queue:

1) Memory is acquired for the new node.
2) Value x is stored in the new node.
3) New node is inserted at the rear end.
4) Special care should be taken for insertion into an empty queue. Both rear and front pointers will point to the only element of the queue.

```
Void que::add(int data)
{   if(front==NULL)
Front=rear=new node(data);
Else{
Rear->next=new node(data);
Rear=rear->next;
}}
```

d. Deletion Function:

```
if(front==NULL)
{
cout<<"\nQueue is empty";
return;
}
```

```
p=front;
front=front->next;
cout<<"\nDeleted job="<<p->data;
if(front==NULL)
rear=NULL;
delete p;
```

e. Print Function:

```
if(front==NULL)
{
cout<<"\nQueue is empty";
return;
}
Cout<<"\nJobs in queue are:";
  for(p=front;p!=NULL;p=p->next)
        cout<<""<<p->data;
```

Algorithm:

Step 1.Start

Step 2. Declare integers i,n,ch,data,arr[MAX],data=10,data1.

Step 3. Declare character ch.

Step 4.Declare nodes *p,*next.

Step 5.Call function create

Step 6.Call function add

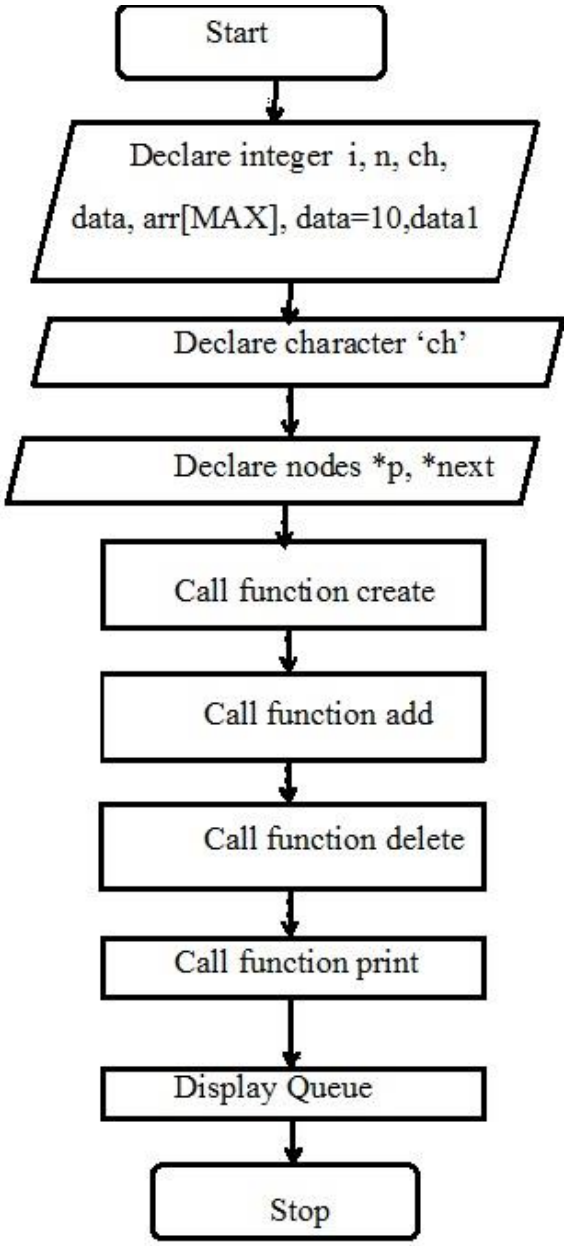
Step 7.Call function delete

Step 8.Call function print

Step 9.Display queue

Step 10.Stop

Flowchart:



Conclusion:

Here,with the help of queue using array and queue using linked list, we successfully simulate a job queue.

Assignment No – 12

Aim: Write a C++ program to simulating double-ended queue by using array.

Objectives:

1. To study dqueue using array

Problem Statement:

A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one- dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

Software Requirements:

Ubuntu Linux14.04,GCC / G++(editor)

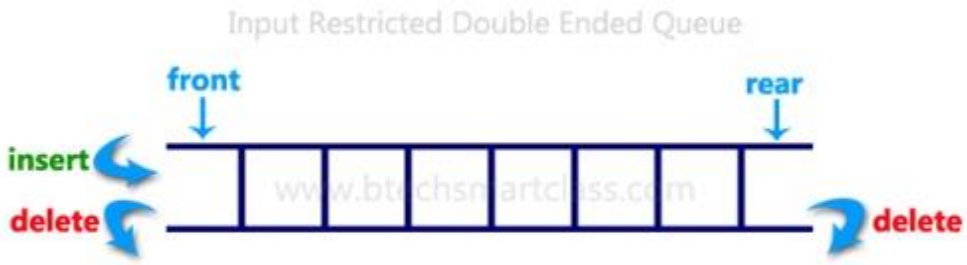
Theory and Concept:

we are going to learn how to create an input and output restricted Deque with the help of array in the data structure?

This differs from the queue abstract data type or First-In-First-Out List (FIFO), where elements can only be added to one end and removed from the other. **This general data class has some possible sub-types**:

1. **An input-restricted Deque** is one where deletion can be made from both ends, but insertion can be made at one end only.
2. **An output-restricted Deque** is one where insertion can be made at both ends, but deletion can be made from one end only.

Input Restricted Deque



Output Restricted Deque

The Below code consists of many functions. Four functions are general, **two display functions**, **two special** and the **main function**.

The implementation starts with the main function and then user choose input or output type of restricted queues. According to the choice, one of the two special functions gets invoked and that function leads the way.

As above explained a little about the operational rules of both types, the user gets the options to operate.

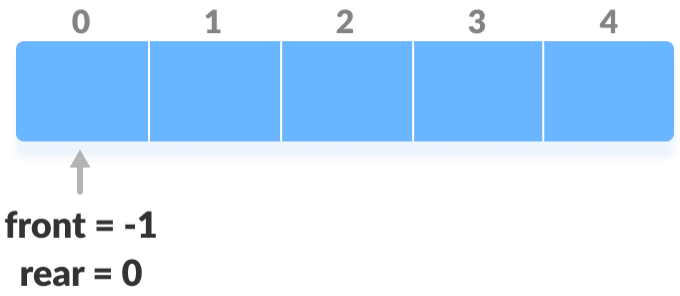
There are four functions insert_left, insert_right, delete_left and delete_right. As the naming specify these functions add or delete to the corresponding sides.

Then we got two display functions for both the different type types of a queue. The Size of array is 5 by default, to change, edit the second line of code.

Before performing the following operations, these steps are followed.

1. Take an array (deque) of size *n*.

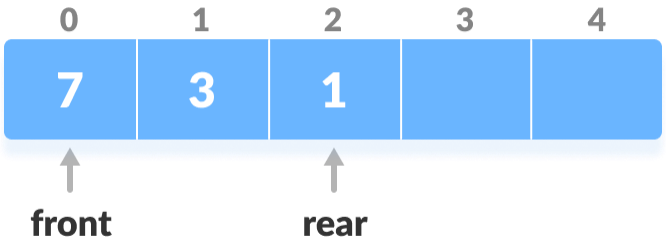
2. Set two pointers at the first position and set `front = -1` and `rear = 0`.



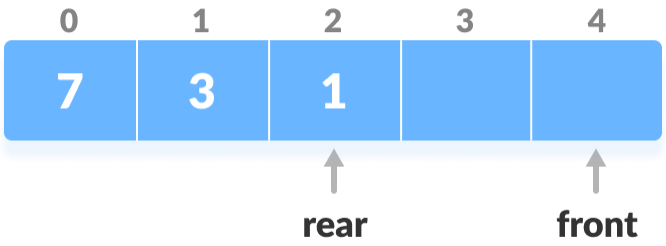
1. Insert at the Front

This operation adds an element at the front.

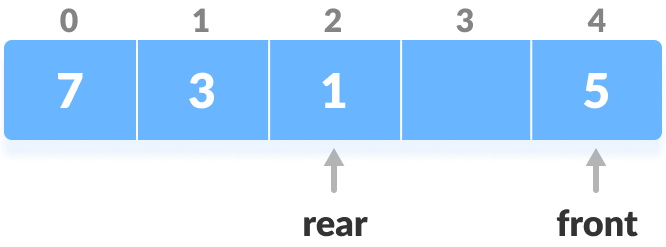
1. Check the position of front.



2.If `front < 1`, reinitialize `front = n-1` (last index).



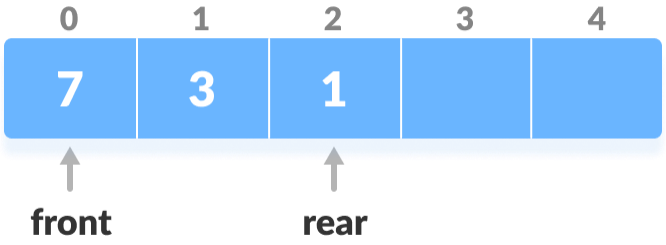
3.Else, decrease *front* by 1.
4 Add the new key *5* into `array[front]`.



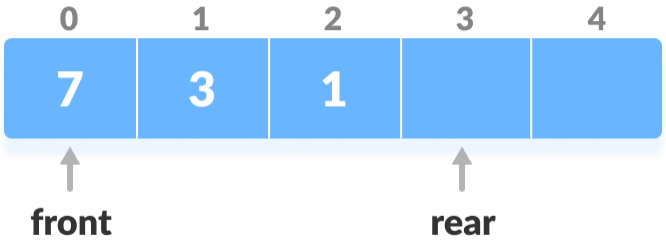
2. Insert at the Rear

This operation adds an element to the rear.

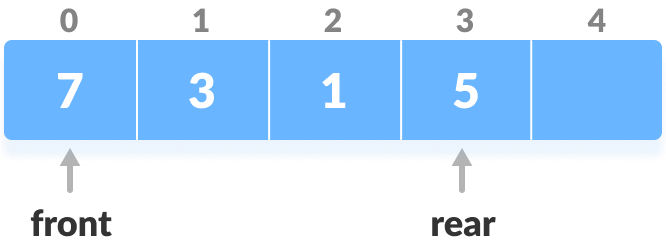
1. Check if the array is full.



2. If the deque is full, reinitialize `rear = 0`.
3.Else, increase *rear* by 1.



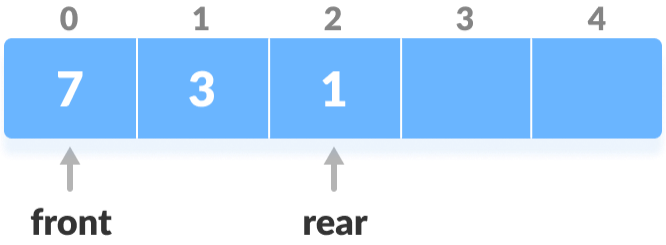
4.Add the new key *5* into `array[rear]`.



3.Delete from the Front

The operation deletes an element from the front.

1. Check if the deque is empty.

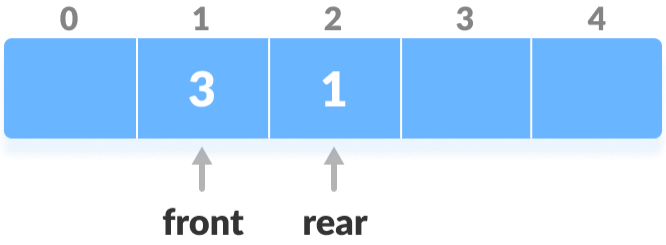


2.If the deque is empty (i.e. `front = -1`), deletion cannot be performed (**underflow condition**).

3 If the deque has only one element (i.e. `front = rear`), set `front = -1` and `rear = -1`.

4 Else if *front* is at the end (i.e. `front = n - 1`), set go to the front `front = 0`.

5 Else, `front = front + 1`.

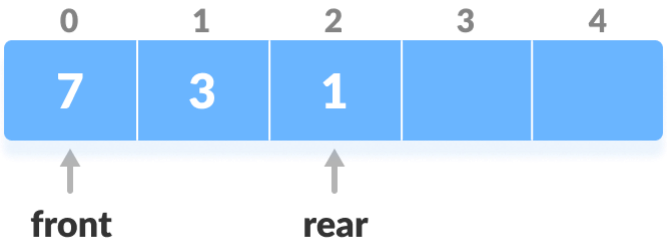


4. Delete from the Rear

This operation deletes an element from the rear.

1. Check if the deque is empty.

2 If the deque is empty (i.e. `front = -1`), deletion cannot be performed (**underflow condition**).
3 If the deque has only one element (i.e. `front = rear`), set `front = -1` and `rear = -1`, else follow the steps below.
4 If *rear* is at the front (i.e. `rear = 0`), set go to the front `rear = n - 1`.



5 Else, `rear = rear - 1`.

5. Check Empty

This operation checks if the deque is empty. If `front = -1`, the deque is empty.

6. Check Full

This operation checks if the deque is full. If `front = 0` and `rear = n - 1` OR `front = rear + 1`, the deque is full.

Algorithm:

Step 1.Start

Step 2. Declare integers i,n,ch,data,arr[MAX],data=10,data1.

Step 3. Declare character ch.

Step 4.Declare nodes *p,*next.

Step 5.Call function create

Step 6.Call function add from front

Step 7 Call function add from rear

Step 8.Call function delete from front

Step 9.Call function delete from rear

Step 10.Call function print

Step 11.Display queue

Step 12.Stop

Conclusion:

Here,we studied operations on dqueue usin array.

Assignment No. - 13

Title: Write a C++ program to implement CIRCULAR queue.

Objectives:-

- To add elements from rear end of Queue.
- To delete elements from front end of Queue.

Problem Statement:-

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..

Software Requirements:-

Fedora with Linux 3.11.10, Eclipse (Editor).

Hardware Requirements :-

Any Processor above Pentium 4.

Theory and Concepts:-

Double Ended Queue:-The word dequeue is a short form of double ended queue.

It is representation of both stack and queue and can be used as stack and queue. In a dequeue, insertion as well as deletion can be carried out either at the rear end or the front end. In practice, it becomes necessary to fix type of operation to be performed on front and rear end. Dequeue can be classified into two types;

A) Input Restricted Dequeue

The following operation are possible in an input restricted dequeue;

1) Insertion of an element at the rear end
2) Deletion of an element from front end
3) Deletion of an element from rear end

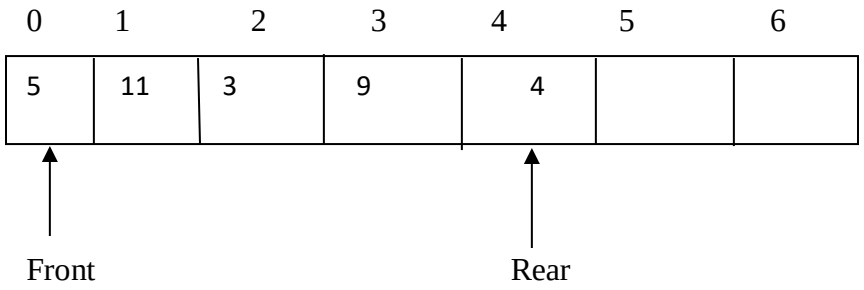
B) Output Restricted Dequeue

The following operation are possible in an output restricted dequeue

1) Deletion of an element from front end
2) Insertion of an element at rear end
3) Insertion of an element at the front end

There are various methods to implement a dequeue

a) Using a circular array
b) Using a singly linked list
c) Using a singly circular linked list
d) Using a doubly linked list
e) Using a doubly circular linked list



a) Initialization

```
voidinit(structque *q)
{
    q->front=-1;
    q->rear=-1;
}
```

b) Print:-

```
void print(structque q)
{
    int i;
    i=q.front;
    while(i!=q.rear)
    {
    cout<<"\t"<<q.arr[i];
       i=(i+1)%MAX;
```

```
    }
    cout<<"\t"<<q.arr[q.rear];
}
```

c) Add Front:-

```
    voidaddf(structque *q,int data)
    {
            if(isempty(*q))
            {
               q->front=q->rear=0;
               q->arr[q->front]=data;
            }
            else
            {
               q->front=(q->front-1+MAX)%MAX;
               q->arr[q->front]=data;
            }
    }
```

D) Add Rear:-

```
    voidaddr(structque *q,int data)
    {
            if(isempty(*q))
            {
               q->front=q->rear=0;
               q->arr[q->rear]=data;
            }
```

```
        else
        {
           q->rear=(q->rear+1)%MAX;
           q->arr[q->rear]=data;
        }
}
```

E) Delete Front:-

```
intdelf(structque *q)
{
        int data1;
        data1=q->arr[q->front];
        if(q->front==q->rear)
        init(q);
else
     q->front=(q->front+1)%MAX;
return data1;
}
```

F) Delete Rear:-

```
intdelr(structque *q)
{
        int data1;
        data1=q->arr[q->rear];
        if(q->front==q->rear)
        init(q);
else
```

```
        q->rear=(q->rear-1+MAX)%MAX;
    return data1;
    }
```

Algorithm:-

Step1:- START

Step 2:- Declare integer array, integer front, rear.

Step 3:- Declare integer data, ch.

Step 4:- Accept integer array from user.

Step 5:- Call functionaddf

Step 6:- Call function addr

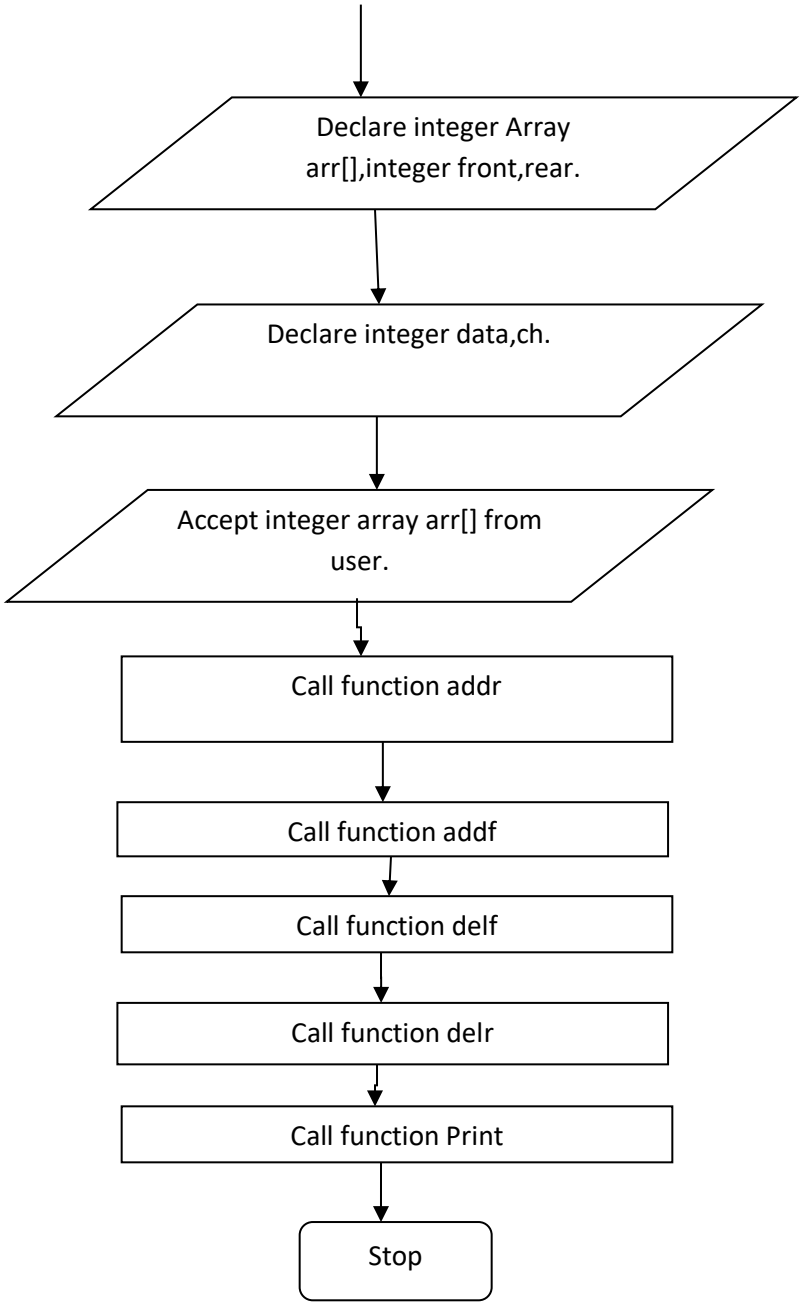
Step 7:- Call function delf

Step 8:- Call function delr

Step 9:-Call function Print.

Stop 10:- Stop

Flowchart:-



Conclusion: By using above code we successfully implemented USING CIRCULAR Queue .