STES's

SINHGAD COLLEGE OF ENGINEERING

Vadgaon(Bk), Pune

Department of Computer Engineering

LABORATORY MANUAL

2022-23

Data Structures Laboratory

(2019 course)

S.E.COMPUTER ENGINEERING

SEMESTER- I

Subject Code: 210246

Teaching Scheme:	**Examination Scheme:**
Practical: 04Hrs/ Week	***Practical: 50 Marks***
Credit: 02	**Term Work: 25 Marks**

Name of Faculty:-

Prof. S.S.Peerzade

Prof. S.P Bholane

Prof. S. S. Pawar

Prof. S. M. Patil

Sinhgad Institutes

Sinhgad College of Engineering
Department of Computer Engineering
Academic Year 2023-24

Second Year of Computer Engineering (2019 Course)
210246: Data Structures Laboratory

Teaching Scheme Practical: 04 Hours/Week	Credit:02	Exam Scheme & Marks Term work: 25 Marks Practical: 50 Marks

Course Objectives:

To understand basic techniques and strategies of algorithm analysis, the memory requirement for various data structures like array, linked list, stack, queue etc using concepts of python and C++ programming language.

Course Outcomes: On completion of the course, learners will be able to

CO1: Use algorithms on various linear data structure using sequential organization to solve reallife problems.

CO2: Analyse problems to apply suitable searching and sorting algorithm to various applications.

CO3: Analyse problems to use variants of linked list and solve various real life problems.

CO4: Designing and implement data structures and algorithms for solving different kinds of problems.

CO-PO Mapping Matrix												
CO/PO	1	2	3	4	5	6	7	8	9	10	11	12
CO 1	1	1	2	1	-	-	-	-	-	-	-	-
CO 2	2	2	2	1	-	-	-	-	-	-	-	-
CO 3	-	2	1	1	-	-	-	-	-	-	-	-
CO 4	1	2	2	1	-	-	-	-	-	-	-	-

Second Year of Computer Engineering (2019 Course)

210246: Data Structures Laboratory

List of Experiments

5	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 a) Selection Sort b) Bubble sort and display top five scores.	28
6	Write **python** program to store 10th class percentage of students in array. Write function for sorting array of floating point numbers in ascending order using radix sort and display top five scores.	34
	Group – C	
7	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 members 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 exist for two divisions. Concatenate two lists.	38
8	Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C++ program to store two sets using linked list. compute and display- a) Set of students who like both vanilla and butterscotch b) Set of students who like either vanilla or butterscotch or not both c) Number of students who like neither vanilla nor butterscotch	49
	Group – D	
9	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.	54
10	Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions: 1. Operands and operator, both must be single character. Input Postfix expression must be in a desired format. Only '+', '-', '*' and '/ ' operators are	58

	expected.	
	Group – E	
11	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.	62
12	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.	69
13	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.	74

Name of Subject Incharge : Dr. M.P. Wankhade

1. Prof. S.S.Peerzade HoD, Dept of Computer Engg
2. Prof. S.P Bholane
3. Prof. S. S. Pawar
4. Prof. S. M. Patil

Assignment No.	1 (GROUP A)
Title	Implement operations on array linear data structure without using built-in functions
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group A- Assignment no: 1

Title: Implement SET operations on array linear data structure without using built-in functions.

Objectives:

1. Understand the concept of linear data structure "ARRAY" to be implemented in Python
2. Understand the basic operation on Array such as Traversing, Insertion, Merging, Searching using python script.

Problem Statement: In second year computer engineering class, group A students play cricket, group B students play badminton and group C students play football.

Write a Python program using functions to compute following: -

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. Number of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton.

(Note- While realizing the group, duplicate entries should be avoided, Do not use SET built-infunctions)

Outcomes: Upon completion of assignment, student will be able to:

1. Design and identify how to use array concept in programming.
2. Understood the concept of linear Data Structures.

Theory Concept in Brief:

A set is a collection of unique data. That is, elements of a set cannot be duplicate.

For example,

Suppose we want to store information about student IDs. Since student IDs cannot be duplicate, we can use a set.

In Python, we create sets by placing all the elements inside curly braces {}, separated by comma.

A set can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

Let's see an example,

```
# create a set of integer type
student_id = {112, 114, 116, 118, 115}
print('Student ID:', student_id)
```

```
# create a set of string type
vowel_letters = {'a', 'e', 'i', 'o', 'u'}
print('Vowel Letters:', vowel_letters)
# create a set of mixed data types
mixed_set = {'Hello', 101, -2, 'Bye'}
print('Set of mixed data types:', mixed_set)
```

Output:

```
Student ID: {112, 114, 115, 116, 118}
Vowel Letters: {'u', 'a', 'e', 'i', 'o'}
Set of mixed data types: {'Hello', 'Bye', 101, -2}
```

In the above example, we have created different types of sets by placing all the elements inside the curly braces {}.

Note: When you run this code, you might get output in a different order. This is because the set has no particular order.

Create an Empty Set

Creating an empty set is a bit tricky. Empty curly braces {} will make an empty dictionary in Python.

To make a set without any elements, we use the set() function without any argument. For example,


```
# create an empty set
empty_set = set()
# create an empty dictionary
empty_dictionary = { }
# check data type of empty_set
print('Data type of empty_set:', type(empty_set))
# check data type of dictionary_set
print('Data type of empty_dictionary', type(empty_dictionary))
```

Output:

```
Data type of empty_set: <class 'set'>
Data type of empty_dictionary <class 'dict'>
```

Here,

empty_set - an empty set created using set()

empty_dictionary - an empty dictionary created using {}

Finally we have used the type() function to know which class empty_set and empty_dictionary belong to.

Duplicate Items in a Set

Let's see what will happen if we try to include duplicate items in a set.

```
numbers = {2, 4, 6, 6, 2, 8}
```

```
print(numbers)   # {8, 2, 4, 6}
```

Here, we can see there are no duplicate items in the set as a set cannot contain duplicates.

Add and Update Set Items

Sets are mutable. However, since they are unordered, indexing has no meaning.

We cannot access or change an element of a set using indexing or slicing. Set data type does not support it.

Add Items to a Set in Python

In Python, we use the add() method to add an item to a set. For example,

```
numbers = {21, 34, 54, 12}
print('Initial Set:',numbers)
# using add() method
numbers.add(32)
print('Updated Set:', numbers)
```

Output:

```
Initial Set: {34, 12, 21, 54}
Updated Set: {32, 34, 12, 21, 54}
```

In the above example, we have created a set named numbers. Notice the line,

```
numbers.add(32)
```

Here, add() adds 32 to our set.

Update Python Set

The update() method is used to update the set with items other collection types (lists, tuples, sets, etc). For example,

```
companies = {'Lacoste', 'Ralph Lauren'}
tech_companies = ['apple', 'google', 'apple']
companies.update(tech_companies)
print(companies)
# Output: {'google', 'apple', 'Lacoste', 'Ralph Lauren'}
```

Here, all the unique elements of tech_companies are added to the companies set.

Remove an Element from a Set

We use the discard() method to remove the specified element from a set. For example,

```
languages = {'Swift', 'Java', 'Python'}
print('Initial Set:',languages)
# remove 'Java' from a set
removedValue = languages.discard('Java')
print('Set after remove():', languages)
```

Output:

Initial Set: {'Python', 'Swift', 'Java'}

Set after remove(): {'Python', 'Swift'}

Here, we have used the discard() method to remove 'Java' from the languages set.

Built-in Functions with Set

Function	Description
all()	Returns `True` if all elements of the set are true (or if the set is empty).
any()	Returns `True` if any element of the set is true. If the set is empty, returns `False`.
enumerate()	Returns an enumerate object. It contains the index and value for all the items of the set as a pair.
len()	Returns the length (the number of items) in the set.
max()	Returns the largest item in the set.
min()	Returns the smallest item in the set.
sorted()	Returns a new sorted list from elements in the set(does not sort the set itself).
sum()	Returns the sum of all elements in the set.

Iterate Over a Set in Python

```
fruits = {"Apple", "Peach", "Mango"}
# for loop to access each fruits
for fruit in fruits:
    print(fruit)
```


Output:

Mango

Peach

Apple

Find Number of Set Elements

We can use the len() method to find the number of elements present in a Set. For example,

```
even_numbers = {2,4,6,8}
print('Set:',even_numbers)
# find number of elements
print('Total Elements:', len(even_numbers))
```

Output:

Set: {8, 2, 4, 6}

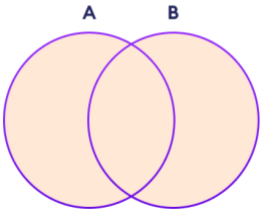
Total Elements: 4

Here, we have used the len() method to find the number of elements present in a Set.

Python Set Operations

Python Set provides different built-in methods to perform mathematical set operations like union, intersection, subtraction, and symmetric difference.

Union of Two Sets



The union of two sets A and B include all the elements of set A and B.

Set Union in Python:

We use the | operator or the union() method to perform the set union operation. For example,

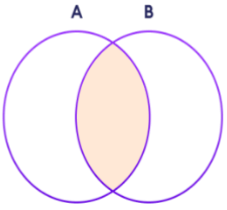
```
# first set
A = {1, 3, 5}
# second set
B = {0, 2, 4}
# perform union operation using |
print('Union using |:', A | B)
# perform union operation using union()
print('Union using union():', A.union(B))
```


Output:

```
Union using |: {0, 1, 2, 3, 4, 5}
Union using union(): {0, 1, 2, 3, 4, 5}
```

Note: A|B and union() is equivalent to A ⋃ B set operation.

Set Intersection



The intersection of two sets A and B include the common elements between set A and B.

Set Intersection in Python

In Python, we use the & operator or the intersection() method to perform the set intersection operation. For example,

```
# first set
```

```
A = {1, 3, 5}
# second set
B = {1, 2, 3}

# perform intersection operation using &
print('Intersection using &:', A & B)
# perform intersection operation using intersection()
print('Intersection using intersection():', A.intersection(B))
```

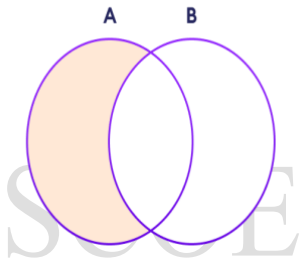
Output:

Intersection using &: {1, 3}

Intersection using intersection(): {1, 3}

Note: A&B and intersection() is equivalent to A ⋂ B set operation.

Difference between Two Sets



The difference between two sets A and B include elements of set A that are not present on set B.

Set Difference in Python

We use the - operator or the difference() method to perform the difference between two sets. For example,

```
# first set
A = {2, 3, 5}
# second set
B = {1, 2, 6}
# perform difference operation using &
print('Difference using &:', A - B)
# perform difference operation using difference()
 print('Difference using difference():', A.difference(B))
```

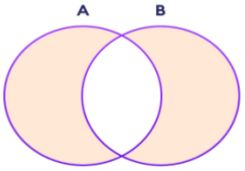
Output:

Difference using &: {3, 5}

Difference using difference(): {3, 5}

Note: A - B and A.difference(B) is equivalent to A - B set operation.

Set Symmetric Difference



The symmetric difference between two sets A and B includes all elements of A and B without the common elements.

Set Symmetric Difference in Python

In Python, we use the ^ operator or the symmetric_difference() method to perform symmetric difference between two sets. For example,

```
# first set
A = {2, 3, 5}
# second set
B = {1, 2, 6}
# perform difference operation using &
print('using ^:', A ^ B)
# using symmetric_difference()
print('using symmetric_difference():', A.symmetric_difference(B))
```

Output:

```
using ^: {1, 3, 5, 6}
using symmetric_difference(): {1, 3, 5, 6}
```

Check if two sets are equal

We can use the == operator to check whether two sets are equal or not. For example,

```
# first set
  A = {1, 3, 5}
# second set
  B = {3, 5, 1}
# perform difference operation using &
if A == B:
    print('Set A and Set B are equal')
else:
    print('Set A and Set B are not equal')
```

Output:

```
Set A and Set B are equal
```

In the above example, A and B have the same elements, so the condition

if A == B

evaluates to True. Hence, the statement print('Set A and Set B are equal') inside the if is executed.

Other Python Set Methods

There are many set methods, some of which we have already used above. Here is a list of all the methods that are available with the set objects:

Method	Description
add()	Adds an element to the set
clear()	Removes all elements from the set
copy()	Returns a copy of the set
difference()	Returns the difference of two or more sets as a new set
difference_update()	Removes all elements of another set from this set
discard()	Removes an element from the set if it is a member. (Do nothing if the element is not in set)
intersection()	Returns the intersection of two sets as a new set
intersection_update()	Updates the set with the intersection of itself and another
isdisjoint()	Returns `True` if two sets have a null intersection
issubset()	Returns `True` if another set contains this set
issuperset()	Returns `True` if this set contains another set
pop()	Removes and returns an arbitrary set element. Raises `KeyError` if the set is empty
remove()	Removes an element from the set. If the element is not a member, raises a `KeyError`
symmetric_difference()	Returns the symmetric difference of two sets as a new set
symmetric_difference_update()	Updates a set with the symmetric difference of itself and another
union()	Returns the union of sets in a new set
update()	Updates the set with the union of itself and others

Conclusion: By this way, students can able to Understand the concept of Array and able to implement this inSet theory Operations using Python Programming.

Questions:

1. What is linear data structure?
2. What are the SET operations?
3. Write an array with three elements.
4. How can Python Programming be useful to implement in SET theory?

Assignment No.	2 (GROUP A)
Title	String Manipulation operations
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group A- Assignment no: 2

Title: String Manipulation operations

Objectives:

1. Write a Python script to perform different operations using arrays.
2. Write a Python script the use standard library functions for 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

Outcomes: Upon completion of assignment, student will be able to:

1. Will be able to understand the concept of char arrays.
2. Will be able to understand how to perform different string operations using arrays.

Theory Concept in Brief:

String Manipulation in Python:

A string is a list of characters in order. A character is anything you can type on the keyboardin one key stroke, like a letter, a number, or a backslash

Strings can have spaces:

"hello world".

An empty string is a string that has 0 characters. Python strings are immutable. Pythonrecognize as strings everything that is delimited by quotation marks (" " or „ „).

String Manipulation

To manipulate strings, we can use some of Pythons built-in methods.

Creation

word = "Hello World"

>>> print word

Output: Hello WorldAccessing

Use [ ] to access characters in a stringword =

"Hello World"

letter=word[0]

>>> print letter

Output: H **Length**

word = "Hello World"

>>> len(word)

Output: 11

Finding word = "Hello World"

>>> print word.count('l') # count how many times l is in thestring

Output: **3**

>>> print word.find("H") # find the word H in the string

Output: **0**

>>> print word.index("World") # find the letters World in the stringOutput: **6**

Count

s = "Count, the number of spaces"

>>> print s.count(' ')

Output: **8 Reversing**

string = "Hello World"

>>> print ''.join(reversed(string))

Output**:** d l r o W o l l e H

Concatenation

To concatenate strings in Python use the "+" operator.

"Hello " + "World" # Output: = "Hello World" "Hello " + "World" + "!"# Output: = "Hello World!"

Copy a String

To Copy a String to another string in Python str1 =

input("Please Enter Your Own String : ")str2 = str1

str3 = str1[:]

print("The Final String : Str2 = ", str2)

print("The Final String : Str3 = = ", str3)

Output:

Please Enter Your Own String: Hello WorldThe

Final String: Str2 = Hello World

The Final String: Str3 = Hello World

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"

print (b[2:5])

Output: llo

Algorithms:

1. Start
2. Accept two string from user i.e. str1
3. Perform to display word with the longest length
4. Perform frequency of occurrence of particular character in the string
5. Calculate the length of string
6. Write a function to check given string are palindrome or not
7. Write a function to find substring
8. Write function to display index of first app2earance of thesubstring
9. Write a function to count the occurrences of each word ina given string
10. Display all string operation
11. End

Flowchart:

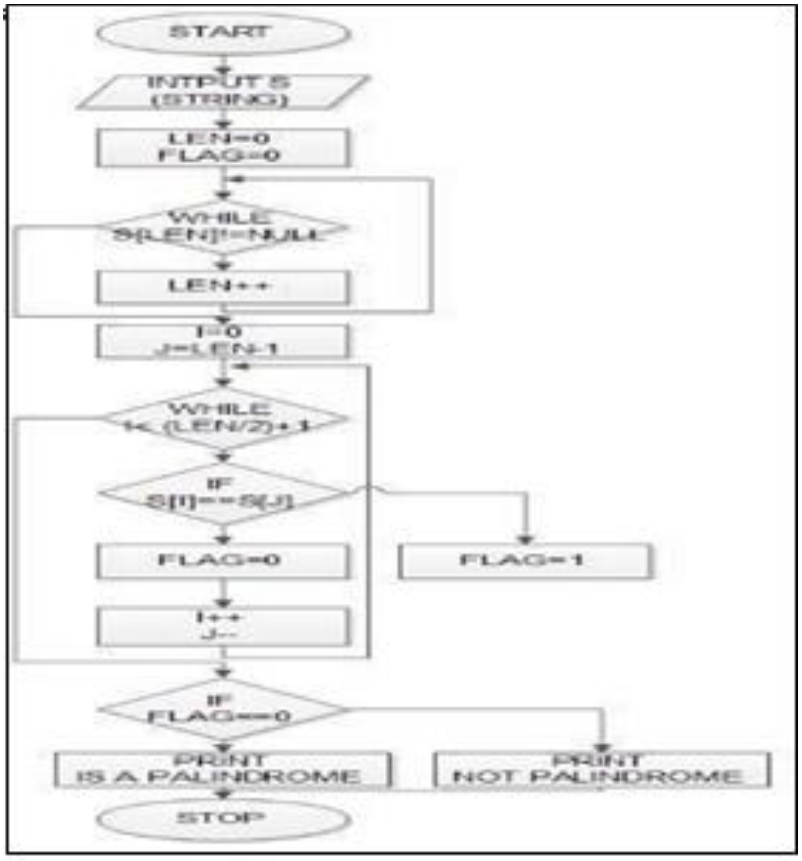


Fig. String is Palindrome or not

Flowchart:

In this way, draw flowchart for remaining functions for string operations

Conclusion:

In this way, students will be able to perform String manipulation operations in python.

Questions:

1. Explain string and character with example.
2. What are the string operations implemented in Python
3. Write a functions to reverse a string
4. Write a Python program to count the number of characters (character frequency) in a string with example
5. How do you remove a given character from String?

Assignment No.	3 (GROUP A)
Title	Matrix Manipulation Operation.
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	

Date	
Signature	

Group A- Assignment no: 3

Title: Matrix Manipulation Operation.

Objectives:

- Compute the transpose of matrix
- Perform addition, subtraction and multiplication of two matrices.

Problem Statement: Write a Python program to compute following computation onmatrix:

a) Addition of two matrices b) Subtraction of two matrices c) Multiplication of two matrices

d) Transpose of a matrix

Outcomes: Upon completion of assignment, student will be able to:

1. Understand OOP concepts in Python
2. Understand to define objects and class in Python

Software Requirements: 64-bit Open source Linux or its derivative, Windows 8/9/10, Programming Tools: G++/GCC compiler, Eclipse IDE.

Hardware Requirements: Intel ® Core™ i3-3220 CPU @3.30 GHz, 4 GB RAM, 500 GBATA HDD

Theory Concept in Brief:

Two-Dimensional Array:

It is a collection of data elements of same data type arranged in rows and columns (that is, in two dimensions).

Declaration of 2 Dimensional Array

Type arrayName[numberOfRows][numberOfColumn];

For example,

int Sales[3][5];

Initialization of Two-Dimensional Array:

A two-dimensional array can be initialized along with declaration. For two-dimensional array initialization, elements of each row are enclosed within curly braces and separated by commas. All rows are enclosed within square braces in python.

A=[[1,2,3],[4,5,6],[7,8,9]]

Referring to Array Elements

To access the elements of a two-dimensional array, we need a pair of indices: one forthe row position and one for the column position. The format is as simple as: name[rowIndex][columnIndex] i.e. A[rowIndex][columnIndex]

Algorithm:

1. Start
2. Input number of rows and columns of first matrix.

3. Input elements of first matrix.
4. Input number of rows and columns of second matrix.
5. Input elements of Second matrix.
6. Function to transpose first matrix i.e. the element at row r column c in the original isplaced at row c column r of the transpose.
7. Function to add, subtract and multiply two matrices

Flowchart: Draw flowchart for above algorithm

Conclusion: By this way, we can perform various operations on matrix successfully

Questions:

1. What are OOPs concepts in Python?
2. How the class and objects define in Python
3. Difference between object-oriented and procedure-oriented programming
4. What are local variables and global variables in Python?
5. What are the key features of Python?

Assignment No.	4 (GROUP B)
Title	Implement Search algorithm Linear, Sentinel, Binary & Fibonacci Search
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group B Assignment no: 4

Title: Implement Search algorithm Linear, Sentinel, Binary & Fibonacci Search

Objectives:

1. Write a Python script to for implementing linear search Sentinel searchtechnique.
2. Write a Python script to for implementing binary search technique.
3. Write a Python script to for implementing Fibonacci search technique.

Problem Statement:

1. 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.
2. 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.

Outcomes:

1. Design and analyze the time and space efficiency of the data structure. Identity the appropriate data structure for given problem.
2. Understand the applications of data structures.

Software & Hardware requirements:

Python 3.4.0/ Jupyter Notebook

Theory- Concept in brief:

Linear Search:

The linear search is a sequential search, which uses a loop to step through an array, starting with the first element. It compares each element with the value being searched for, and stops when either the value is found or the end of the array is encountered. If the value being searched is not in the array, the algorithm will unsuccessfully search to the end of the array. Since the array elements are stored in linear order searching the element in the linear order make it easy and efficient. The search may be successful or unsuccessfully. That is, if the required element is found then the search is successful otherwise it is unsuccessful.

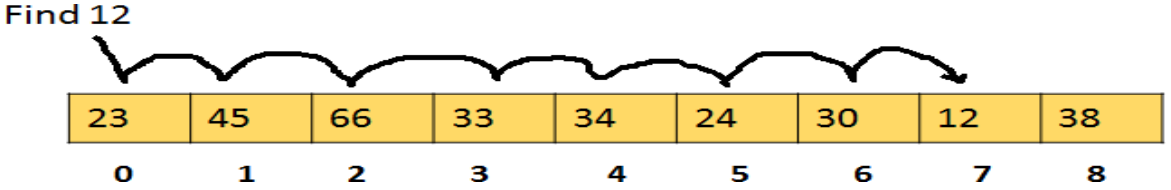


Fig. Linear Search Example

Linear Search Algorithm:

Consider an integer type array A with size n. So list of elements from thatarray are,

A[0], A[1], A[2], A[3], ..., A[n-1]

1. Declare and initialize one variable which contents the number to be searchin an array A. (variable key is declared)
2. Start Comparing each element from array A with the key LOOP: A[size] ==key
   Repeat step no 2 while A[size] key
3. if key is found, display the location of element(index+1) or else display message KEY NOT FOUND
4. Terminate the program successfully

Sentinel Search:

In this search, the last element of the array is replaced with the element to be searched 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).

Sentinel Search Algorithm:

Algorithm SentinelSearch (list, last, target, location)

1. Set list[last+1] to target
2. Set looker to 0
3. Loop(target not equal list[looker])
   Increment looker
4. End loop
5. If(looker<=last)
   - Set found to true
   - Set location to looker
6. Else
   - Set found to false
   - Set location to last
7. End if
8. Return found

Binary search is an searching algorithm which is used to find element from thesorted list".

Advantages of Binary Search

1. Binary search is optimal searching algorithms
2. Excellent time efficiency
3. Suitable for large list.
4. Faster because no need to check all element.
5. Most suitable for sorted array
6. It can be search quickly Time complexity O(log n)

Disadvantages of Binary Search

1. Element must be sorted
2. Need to find mid element
3. Bit more complicated to implement and test
4. It does not support random access.
5. Key element require to compare with middle.

Fibonacci Search is a comparison-based technique that uses Fibonacci numbersto search an element in a sorted array.

Fibonacci Numbers are recursively defined as F(n) = F(n-1) + F(n-2), F(0) = 0,F(1) = 1.

First few Fibonacci Numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,….

Similarities with Binary Search:

Works for sorted arrays

A Divide and Conquer Algorithm.Has Log n time complexity.

Differences with Binary Search:

Fibonacci Search divides given array in unequal parts Binary Search uses division operator to divide range. 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.

Algorithm:

Binary Search Algorithm

Assume that two variables are declared, variable first and last, they denotes beginning and ending indices of the list under consideration respectively.

Step 1. Algorithm compares key with middle element from

list (A[middle] == key), if true go to step 4 or else go to next.

Step 2. if key < A[middle], search in left half of the list or else go tostep 3

Step 3. if key > A[middle], search in right half of the list or go to step 1Step 4.display the position of key else display message “NOT FOUND”

Example:

Consider a sorted list a[] with 9 elements and the search key is 31.

0	1	2	3	4	5	6	7	8
11	23	31	33	65	68	71	89	100

Let the search key = 31.

First low = 0, high = 8, mid = (low + high) = 4 a[mid] = 65 is the centre element, but 65 > 31.

So now high = mid - 1= 4 - 1 = 3, low = 0, mid = (0 + 3) / 2 = 1

a[mid] = a[1] = 23, but 23 < 31.

Again low = mid +1 = 1 +1 =2, high = 3, mid = (2 + 3) /2 = 2

a[mid] = a[2] = 31 which is the search key, so the search is successful.

Fibonacci Search Algorithm

Let arr[0..n-1] be the input array and element to be searched be x.

Step 1:Find the smallest Fibonacci Number greater than or equal to n. Let this number be fibM [m‟th Fibonacci Number]. Let the two Fibonacci numbers preceding it be fibMm1 [(m-1)‟th Fibonacci Number] and fibMm2 [(m-2)‟th Fibonacci Number].

Step 2: While the array has elements to be inspected:

Compare x with the last element of the range covered by fibMm2

If x matches, return index

Else If x is less than the element, move the three Fibonacci variables two Fibonacci down, indicating elimination of approximately rear two-third of the remaining array.

Else x is greater than the element, move the three Fibonacci variables one Fibonacci down. Reset offset to index. Together these indicate elimination of approximately front one-third of the remaining array.

Step 3: Since there might be a single element remaining for comparison check iffibMm1 is 1. If Yes, compare x with that remaining element. If match, return index.

Example:

Consider a list a[] with 11 elements and the search element is 85.n = 11

Index	0	1	2	3	4	5	6	7	8	9	10
a[i]	10	22	35	40	45	50	80	82	85	90	100

Smallest Fibonacci number greater than or equal to 11 is 13. fibMm2 = 5,fibMm1 = 8, fibM = fibMm1 + fibMm2 = 13 Initialize offset = 0

Check the element at index i = min(offset + fibMm2, n)

fibMm2	fibMm1	fibM	Offset	i=min(offset+fibln, N)	A[i]	Consequence
5	8	13	0	5	45	Move one down,reset offset to i
3	5	8	5	8	82	Move one down, reset offset to i
2	3	5	8	10	90	Move Two Down
1	1	2	8	9	85	Return i

Flowchart:

Binary Search

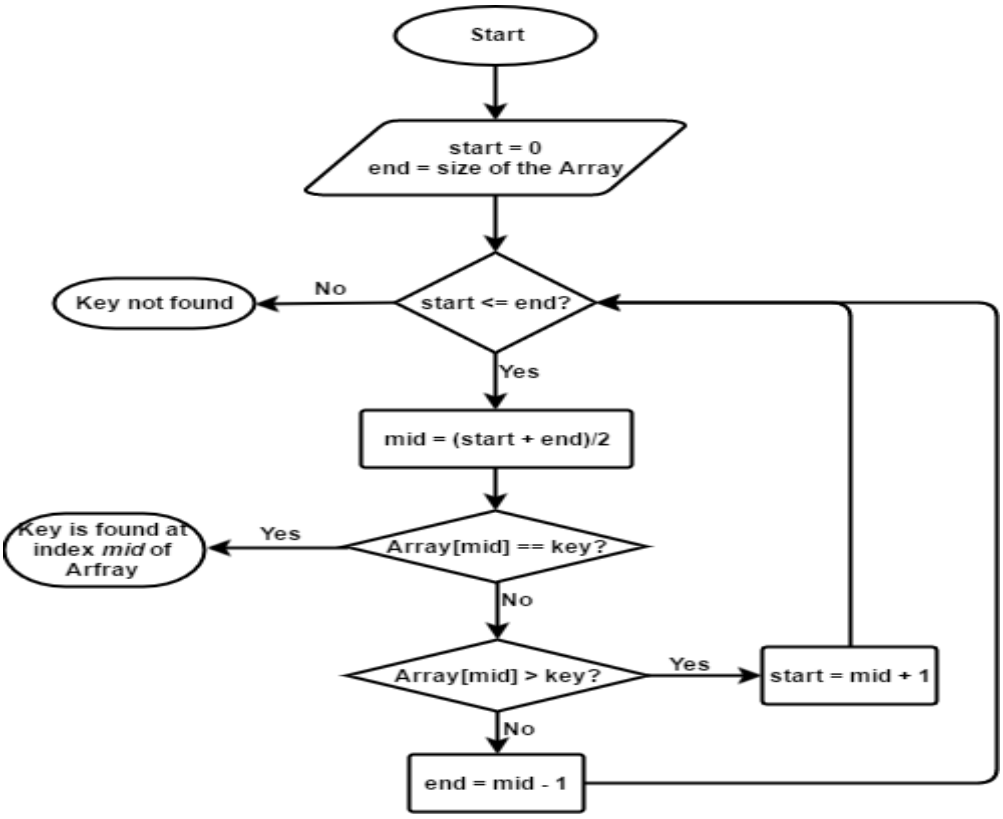


Fig. Binary Search Flowchart

Draw Flowchart for Linear, Sentinel andFibonacci Search

Conclusion/analysis:

Linear, Sentinel, Binary & Fibonacci Search implemented successfully.

Assignment No.	5 (GROUP B)
Title	Selection & Bubble Sort
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group B- Assignment no: 5

Title: Selection & Bubble Sort

Objectives:

1. Write Python script for implementing selection sort techniques to arrange alist of floating point numbers in ascending order.
2. Write Python script for implementing Bubble sort techniques to arrange alist of floating point numbers in ascending order.

Problem Statement:

Write a python program to store first year percentage of students in array. Writefunction for sorting array of floating point numbers in ascending order using

- Selection Sort
- Bubble sort and display top five scores.

Outcomes:

Design and analyze the time and space efficiency of the data structure. Identity the appropriate data structure for given problem.

Software & Hardware requirements:

Python 3.4.0 / Jupyter Notebook

Theory- Concept in brief:

Selection Sort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two sub-arrays in a given array. The sub-array which is already sorted. Remaining subarray which is unsorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub-array is picked and moved to the sorted sub-array.

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.

Algorithm:

Selection Sort Algorithm

Algorithm selectionSort (low, high)

```
{ //a[low : high] is an array of size n
 i=0, j=0, temp=0, ;
for i: =low to high do
{
minindex = i;
for j: =i+1 to high do
{
if( a[j] < a[minindex] ) then minindex := j;
}
temp := a[i];
a[i] := a[minindex]; a[minindex] := temp;
}
}
```

Example: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. In every iteration of selection sort, the minimum element from the unsorted sub-array is picked and moved to the sorted sub-array.

Consider a list a = [64, 25, 12, 22, 11]

Index	**0**	**1**	**2**	**3**	**4**	**Remarks**
List	64	25	12	22	11	Find the minimum element in a[0 .. 4] and place it at beginning.
Iteration 1	11	25	12	22	64	Find the minimum element in a[1 .. 4] and place it at beginning of a[1 .. 4]
Iteration 2	11	12	25	22	64	Find the minimum element in a[2 .. 4] and place it at beginning of a[2 .. 4]
Iteration 3	11	12	22	25	64	Find the minimum element in a[3 .. 4] and place it at beginning of a[3 .. 4]
Iteration 4	11	12	22	25	64	Finally the list is sorted.

Bubble Sort Algorithm

```
Algorithm bubblesort ( x[], n)
{ // x[1:n] is an array of n elements
for i := 0 to n do
{
```

```
for j := 0 to n–i-1 d0
{
if (x[j] > x[j+1])
{
temp =  x[j]; x[j] = x[j+1]; x[j+1] = temp;
}
}
}
```

Example: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are not in order.

First Pass

5	1	4	2	8	Compare the first two elements, and swaps since 5 > 1
1	5	4	2	8	Compare 5 and 4 and swap since 5 > 4
1	4	5	2	8	Compare 5 and 2 and swap since 5 > 2
1	4	2	5	8	Compare 5 and 8 and since 5 < 8, no swap.

Second Pass

1	4	2	5	8	Compare the first two elements, and as 1 < 4, no swap.
1	4	2	5	8	Compare 4 and 2, swap since 4 > 2
1	2	4	5	8	Compare 4 and 5, no swap.
1	2	4	5	8	Compare 5 and 8 and no swap.

Third Pass

1	2	4	5	8	Compare the first two elements and no swap.
1	2	4	5	8	Compare 2 and 4, no swap.
1	2	4	5	8	Compare 4 and 5, no swap.
1	2	4	5	8	Compare 5 and 8, no swap.

Flowchart:

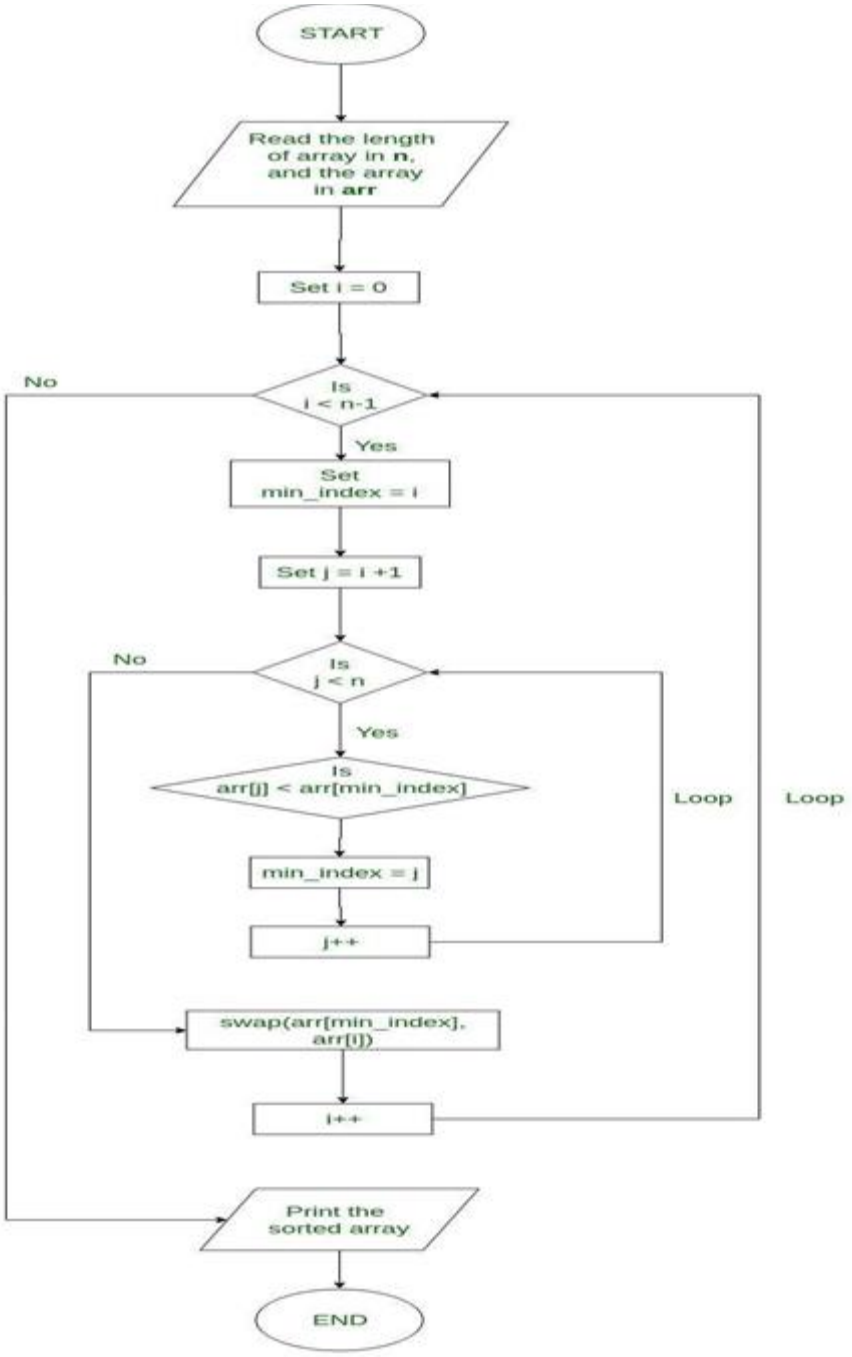


Fig. Selection Sort Flowchart

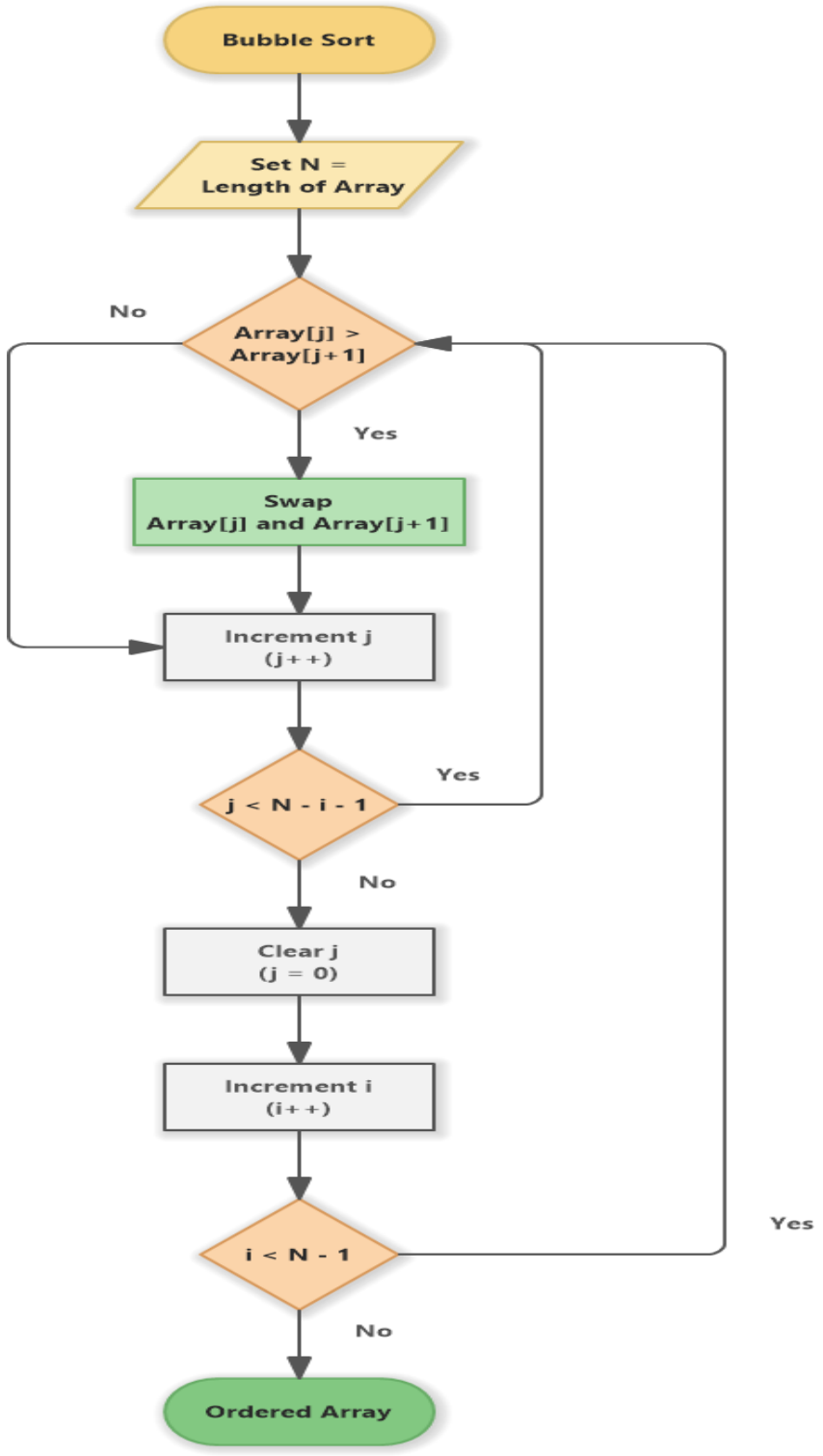


Fig. Bubble Sort Flowchart

Conclusion/analysis:

Successfully sorted first year percentage of students using Selection sort & bubble sorttechniques.

Assignment No.	6 (GROUP B)
Title	Radix Sort
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group B Assignment No 6

Title: Radix Sort

Objectives:

Write Python script for implementing Radix sort techniques to arrange a list of floating point numbers in ascending order.

Problem Statement:

Write **python** program to store 10th class percentage of students in array. Write function for sorting array of floating point numbers in ascending order using radix sort and display top five scores

Outcomes:

Design and analyze the time and space efficiency of the data structure. Identity the appropriate data structure for given problem.

Software & Hardware requirements:

Python 3.4.0 / Jupyter Notebook

Theory- Concept in brief:

Radix Sort

Radix sort is a small method that many people intuitively use when alphabetizing a large list of names. Specifically, the list of names is first sorted according to the first letter of each name, that is, the names are arranged in 26 classes.

Intuitively, one might want to sort numbers on their most significant digit. However, Radix sort works counter-intuitively by sorting on the least significant digits first. On the first pass, all the numbers are sorted on the least significant digit and combined in an array. Then on the second pass, the entire numbers are sorted again on the second least significant digits and combined in an array and so on.

Algorithm:

```
radixSort(array)
  d
```

```
countingSort(array, d)
  max
  <-
  find
```

Example

Input	1st Pass	2nd Pass	3rd Pass
329	720	720	329
457	355	329	355
57	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Flowchart:

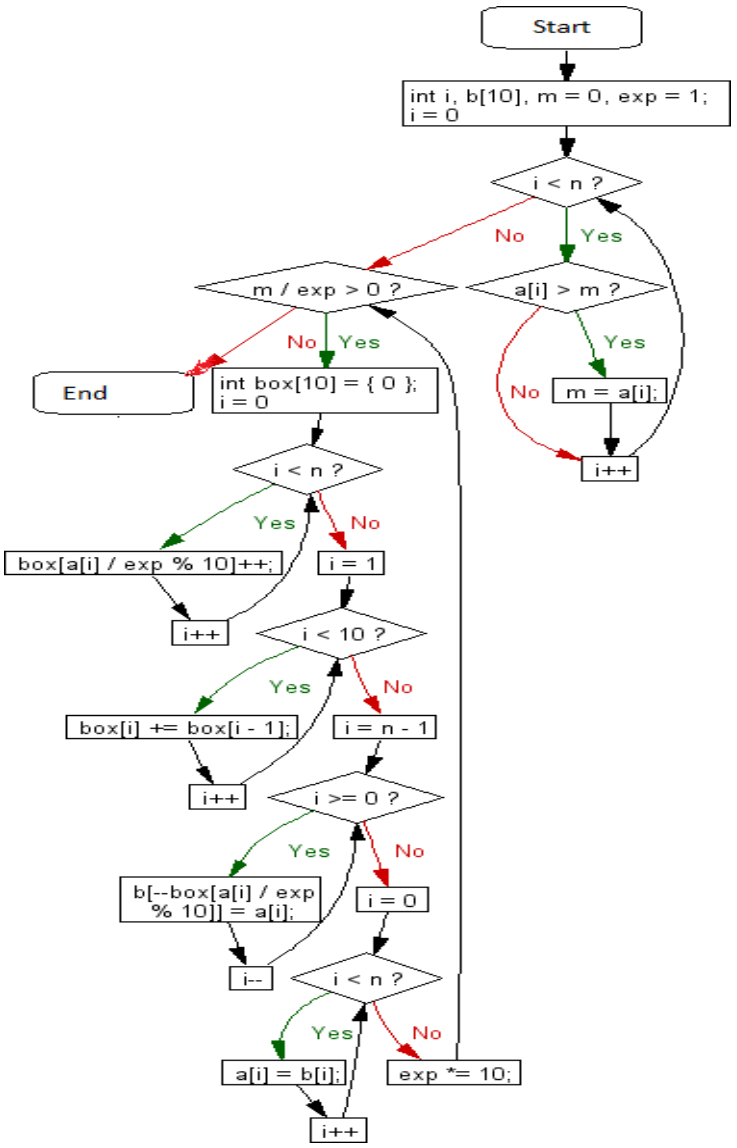


Fig. Radix Sort Flowchart

Conclusion/analysis: Radix Sort Implemented Successfully

Assignment No.	7 (GROUP C)
Title	Maintain club members information using singly linked list
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group C- Assignment no: 7

Title: Write C++ program to maintain club members information using singly linked list

Objectives:

1. To understand the concept and features of singly linked list data structure.
2. To maintain club member's information by performing different operations like add, delete, display, concatenate and total count of nodes on 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 members information using singly linked list. Store student PRN andName. 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.

Outcomes: Upon the completion of Single Linked list operations, the student will be able to:

1. Design the singly linked list data structure
2. Understand to apply operations like insert and delete the node at beginning, at end and after the given node.
3. Compare different implementations of Singly linked list data structures and to recognize the advantages and disadvantages of them.
4. Choose the appropriate list declared for two divisions and design algorithm for specified application.

Software Requirements: 64-bit Open source Linux or its derivative, Windows 8/9/10, Programming Tools: G++/GCC compiler, Eclipse IDE.

Hardware Requirements: Intel ® Core™ i3-3220 CPU @3.30 GHz, 4 GB RAM, 500 GB ATA HDD

Theory Concept in Brief:

A linked list is a sequence of data structures, which are connected together via links. A linked list consists of nodes where each node contains a data field and a reference (link) to the next nodein the list.

Linked List Representation

Fig. Linked List Representation

Linked list can be visualized as a chain of nodes, where every node points to the next node.Following are the important points to be considered.

- Linked List contains a link element called first.
- Each link carries a data field(s) and a link field called next which contains the addressof the next node in the memory.
- The last node of the list contains pointer to the null.

Types of Linked list:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Singly Linked List:

Fig. Single Linked List Representation

It is the most common. Each node has data and a pointer to the next node.

Doubly Linked List:

We add a pointer to the previous node in a doubly-linked list. Thus, we can go in either direction: forward or backward

Fig. Double Linked List Representation

Circular Linked List:

A circular linked list is a variation of a linked list in which the last element is linked to the first element. This forms a circular loop.

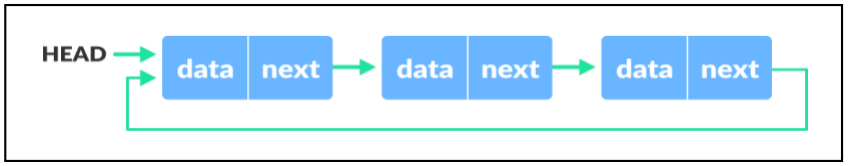


Fig. Double Linked List Representation

Singly Linked list:

Singly linked list can be defined as the collection of ordered set of elements. The number of elements may vary according to need of the program. A node in the singly linked list consist of two parts: data part and link part. Data part of the node stores actual information that is to be represented by the node while the link

part of the node stores the address of its immediate successor.

Operations on Singly Linked List

There are various operations which can be performed on singly linked list. A list of all suchoperations is given below.

Node Creation

```
struct node { int data;

struct node *next;
};
struct node *head, *ptr;
ptr = (struct node *)malloc(sizeof(struct node *));
```

Insertion

The insertion into a singly linked list can be performed at different positions. Based on the position of the new node being inserted, the insertion is categorized into the following categories.

SN	Operation	Description
1	Insertion atbeginning	It involves inserting any element at the front of the list. The new node can be inserted as the head of the list.
2	Insertion at end ofthelist	It involves insertion at the last of the linked list. The new node can be inserted as the only node in the list or it can be inserted as the last one.
3	Insertion afterspecified node	It involves insertion after the specified node of the linked list. We need to skip thedesired number of nodes in order to reach the node after which the new node will be inserted. .

Deletion and Traversing

The Deletion of a node from a singly linked list can be performed at different positions. Based on the position of the node being deleted, the operation is categorized into the following categories.

SN	Operation	Description
1	Deletion	It involves deletion of a node from the beginning of the list.
2	Deletion at th	It involves deleting the last node of the list. The list can either be empty or full. This requires traversing through the list.
3	Deletion afterspecified node	It involves deleting the node after the specified node in the list. we need to skip thedesired number of nodes to reach the node after which the node will be deleted. This requires traversing through the list.
4	Traversing	In traversing, we simply visit each node of the list at least once in order to performsome specific operation on it, for example, display the total number of node present in the list.
5	Searching	In searching, we match each element of the list with the given element. If the element is found on any of the location then location of that element is returnedotherwise null is returned. .

A three-member singly linked list can be created as:

```
/* Initialize nodes */ struct node *head;
struct node *one = NULL; struct node *two = NULL;
struct node *three = NULL;
/* Allocate memory */
one=malloc(sizeof(structnode));
two=malloc(sizeof(structnode));
three=malloc(sizeof(struct node));
/* Assign data values */ one->data = 1;
two->data = 2;
three->data = 3;
/* Connect nodes */ one->next = two; two->next = three;
three->next = NULL;
/* Save address of first node in head */ head = one;
```


Algorithm:

1. Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list and data contains PRN number and name of the students.
2. Create another class list which has two attributes: head and tail of a node and declares functions for Singly Linked list operations to be implemented.

3. InsertNode() will add a new node to the list:

- Create a new node.
- It first checks, whether the head is equal to null which means the list is empty.
- If the list is empty, both head and tail will point to the newly added node.
- If the list is not empty, the new node will be added to start of the list such that head's next will point to a newly added node. This new node will become the new head of the list.
- If the list is not empty, the new node will be added to end of the list such that tail's next will point to a newly added node. This new node will become the new tail of the list.
- Enter the data to insert after node, traverse through list till last node, such that a newly created node tails next point to previous node tail and make the node inserted in the list.

4. countNodes() will count the nodes present in the list:

Define a node current which will initially point to the head of the list.

- Declare and initialize a variable count to 0.
- Traverse through the list till current point to null.
- Increment the value of count by 1 for each node encountered in the list.

5. display() will display the nodes present in the list:

Define a node current which will initially point to the head of the list.

- Traverse through the list till current points to null.
- Display each node by making current to point to node next to it in each iteration.

6. deleteNode() will delete a new node to the list:

1) Create a new node.
2) It first checks, whether the head is equal to null which means the list is empty.

i) If the list is not empty then head and tail point to first node and make tail‟s nextto null means delete the first node.

ii) If the list is not empty then traverse through list till last node, head and tail point to last nod and make tail‟s next to null means delete the last node.

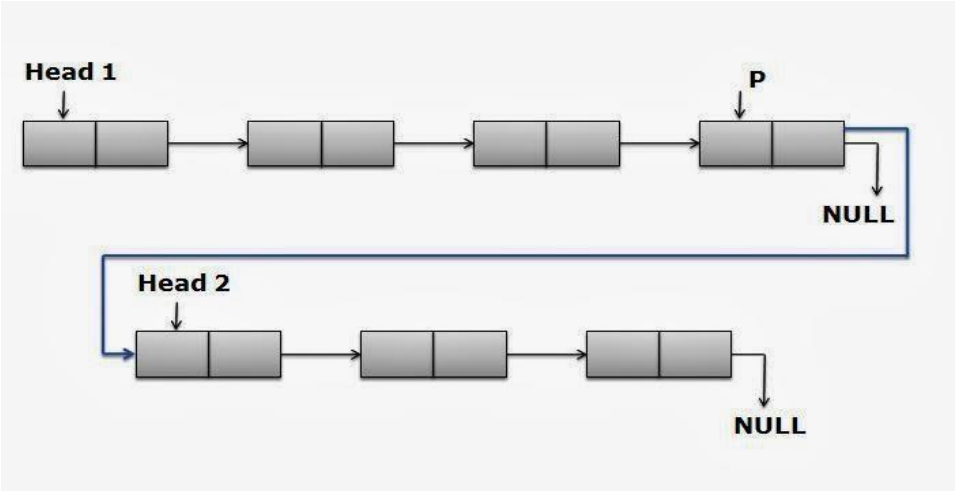
iii) If the list is not empty then enter the number to delete, traverse through list till last node, if data found at particular node then change next pointers to exclude the node from the list and next nodes tail to null means delete the data after position of node.

7. Concatenate() will concatenate two linked lists are referenced by head1 and head2 respectively.

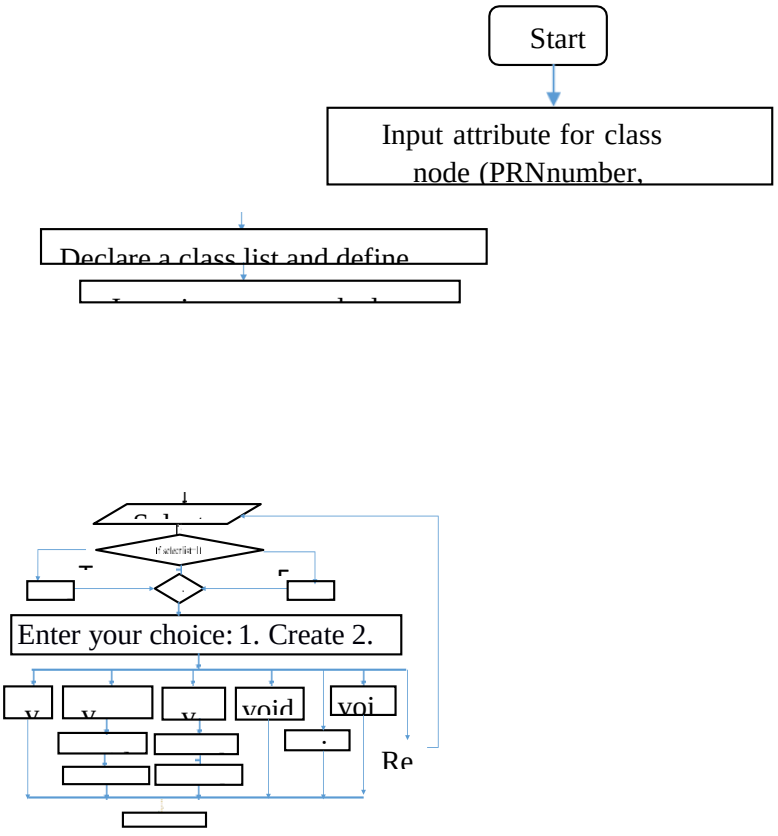
1) If the first linked list is empty then return head2.

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, sayp.
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 pointed by
6) p. Return head1.



Flowchart:



Test Cases: **SINGLY-LINKED LIST**

Preconditions:

Node A points to node C ; [NODE A] >[NODE C]Node B is an insertion node; Insert [NODE B]

Test cases:

TC1) You can insert [NODE B] after [NODE A] TC2) You can not insert [NODE B] before [NODE A] TC3) [NODE A] points to [NODE B] after insertion TC4) [NODE B] points to [NODE C] after insertion TC5) You can remove [NODE B]

TC6) [NODE A] points to [NODE C] after removal

Mathematical Model: A mathematical model is a description of a system using mathematical concepts andlanguage. The process of developing a mathematical model is termed mathematical modeling.

$M = \{I, F, O, S_c, F_r\}$

Where M be the system defines for Singly linked list to maintain club members information Ibe the input of the students such as PRN number and name of the student

F be the functions declared to implement singly linked list operations

O be the output define to display members of the club after implementing singly linked listoperations S_c be the success rate of singly linked list implementation F_r be the failure rate of singly linked list implementation

$I = \{I_1, I_2\}$ where I_1 be the PRN number of student and I_2 be the name of the student $F = \{F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}\}$

Where F_1 be function declared for creation of node in the listF_2 be the function declared todisplay the list

F_3 be the function declared to insert the President at the beginning of the list

F_4 be the function declared to insert the member after any member present in the listF_5 be the function declared to insert the Secretary at the end of the list

F_6 be the function declared to delete the President at the beginning of the list

F_7 be the function declared to delete the member after any member present in the list F_8 bethe function declared to delete the Secretary at the end of the list

F_9 be the function declared to count the total number of members present in the listF_{10} be the function declared to concatenate two list of two divisions.

Start defines first node

$F_1 = \{$if start==NULL then $F_1 \epsilon$ N(I)$\}$ where N be the node with $N=\{I, L\}$ means $N \epsilon I$ and $N \rightarrow L$(link)$F_2 =$ {if start==NULL then List is EMPTY; otherwise $F_2 \epsilon$ O display members of the club}

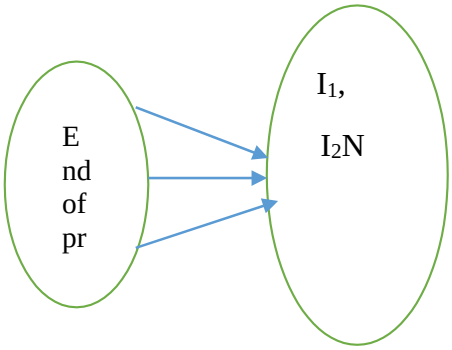
$F_3 = \{$if start==NULL then $F_3 = F_1$ otherwise $F_3 \epsilon$ I$\}$

$F_4 = \{$if $I_1 = F_4(I_1)$ then $F_4 \epsilon$ I and insert member after any member present in the list$\}$

F_5 = {if (start==NULL) then F_5= F_1 otherwise $F_5\epsilon$ I and N→(L!=NULL) then $F_5\epsilon$ N(I)}F_6 = {if (start==NULL) then club is empty otherwise N→L=NULL delete President} F_7 = {if I_1= $F_7(I_1)$ then N(I)= N→L and N→L=NULL delete the member}

F_8 = { if(start==NULL) then club is empty otherwise traverse to last node N→L=NULL delete Secretary}F_9 = {Initially count=0; Traverse to last node till N!=NULL and count++} F_{10} = {(L1+L2) where L1 be first list and L2 be second list; make second list of head node= NULL}

O = O ϵ {F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}}



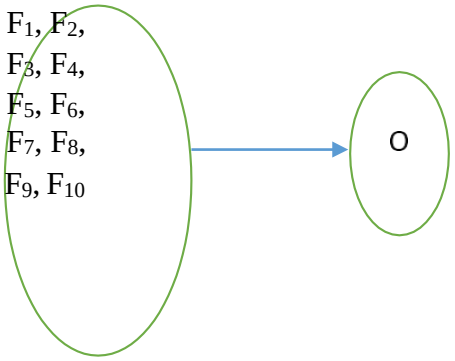


Figure1: set of Inputs belongs to functions **Figure2: set of Functions delivers the outputs**

S_c= Success rate= Node is inserted or deleted at assigned location {F_3, F_4, F_5, F_6, F_7, F_8} ϵ N(I)

F_r= Failure rate= If node is not found in particular location; N!=NULL (list is empty); If concatenation doesn't occur, F_{10} doesn't belong to L2 then second list of head node is not assigned to first list

Conclusion: By this way, we can maintain club members information using singly linked list.

Questions:

1. What is a Linked list?
2. Can you represent a Linked list graphically?
3. How many pointers are required to implement a simple Linked list?
4. How many types of Linked lists are there?
5. How to represent a linked list node?
6. Describe the steps to insert data at the starting of a singly linked list.
7. How to insert a node at the end of Linked list?
8. How to delete a node from linked list?
9. How to reverse a singly linked list?
10. What is the difference between singly and doubly linked lists?
11. What are the applications that use Linked lists?
12. What will you prefer to use a singly or a doubly linked lists for traversing through a listof elements?

Assignment No.	8 (GROUP C)
Title	Set operations using linked list
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group C Assignment No 8

Title: Set operations using linked list.

Objectives: To understand set operation. Representation and implementation of operation of sets using linked list

Problem Statement:

Second year Computer Engineering class, set A of students like Vanilla Ice- cream and set B of students like butterscotch ice-cream. Write C++ program to store two sets using linked list. compute and display-

a) Set of students who like both vanilla and butterscotch
b) Set of students who like either vanilla or butterscotch or not both
c) Number of students who like neither vanilla nor butterscotch

Outcomes:

a. List of students who like either vanilla or butterscotch or both
b. List of students who like both vanilla and butterscotch
c. List of students who like only vanilla not butterscotch
d. List of students who like only butterscotch not vanilla
e. Number of students who like neither vanilla nor butterscotch

Software & Hardware requirements:

Linux operating system Eclipse IDE with g++ compiler

Theory- Concept in brief:

Set:

a set is a collection of objects which are called the members or elements of that set. If we have a set we say that some objects belong (or do not belong)to thisset, are (or are not) in the set.

Examples: the set of students in this room; the English alphabet may beviewed as the set of letters of the English language; the set of natural numbers Operations on sets:

1. Union: The union of A and B, written A ∪ B , is the set whose elementsare just the elements of A or B or of both.
2. Intersection: The intersection of A and B , written A ∩ B , is the set whose elements are just the elements of both A and B.
3. Difference: Another binary operation on arbitrary sets is the difference — Aminus B‖ , written A – B, which ‗subtracts„ from A all elements which are in

4. B. [Also called relative complement: the complement of B relative to A.]

5. Complement: This operation is creating a set A, which is the set consisting of everything not in A

Algorithm:

Algorithm difference(struct node *head1, struct node *head2) Precondition: Accept the two sets of name in the form of linked list. **Post condition**: Difference of two sets

Return: head node of the resultant linked list(Difference)

1. Set p=head1, i.e. head of 1st linked list

2. Initialize the 3rd linked list as empty i.e. head3=NULL

3. while(p!=NULL)

a. Set flag=0

b. Set q=head2, i.e. head of 2ndlinked list

c. while(q!=NULL)

i. if(strcmp(p->name,q->name)==0)

I. Set flag to 1 and go to step d

ii. else

I.Move q to the next node, q=q->next

d. if(flag!=1)

i. if(head3==NULL)

I. Allocate memory for New node

II. strcpy(New->name,p->name)

III. Set r=New and r->next=NULL

IV. Make New node as head node of resultant linked list

ii.else

I. Allocate memory for New node

II. r->next=New

III. strcpy(New->name,p->name)

IV. Set r=r->next and r->next=NULL

e. Move p to the next node, p=p->next

4. return head3 i.e. head of resultant linked list

5. Stop

Algorithm Intersection(struct node *head1, struct node *head2) Precondition: Accept the two sets of name in the form of linked list. **Post condition**: Intersection of two sets

Return: Nil

1. Set p=head1, i.e. head of 1st linked list
2. Initialize the 3rd linked list as empty i.e. head3=NULL
3. while(p!=NULL)
   - **a.** Set q=head2, i.e. head of 2nd linked list
   - **b.** while(q!=NULL)
     - **i.i** f(strcmp(p->name,q->name)==0)
       - **I.** if(head3==NULL)
         1. Allocate memory for New node
         2. strcpy(New->name,p->name)
         3. Set r=New and r->next=NULL;
         4. Make new node as head of linked list
       - **II.** else
         1. Allocate memory for New node
         2. r->next=New
         3. strcpy(New->name,p->name)
         4. Set r=r->next and r->next=NULL
       - **III.** Go to step c
     - **ii.** Move q to the next node, q=q->next
   - **c.** Move p to the next node, p=p->next
4. Call display(head3) function to display resultant linked list containing intersection
5. Stop

Algorithm Union(struct node *head1,struct node *head2) Precondition: Accept the two sets of name in the form of linked list. **Post condition**: Union of two sets

Return: head node of the resultant linked list(Union)

1. Set q=head2, i.e. head of 2nd linked list
2. Copy all contents of first linked list in third, head3=head1
3. Set r=head3, i.e. head of 3rd linked list

4. while(r->next!=NULL) Move r to the next node, r=r->next

5. while(q!=NULL)

a. Set p=head1, i.e. head of 1st linked list

b. Initialize flag to 0

c. while(p!=NULL)

i. if(strcmp(p->name,q->name)==0) Set flag to 1 and go to step d

ii. else Move p to the next node, p=p->next

d. if(flag==0)

i. Allocate memory for New node

ii. r->next=New

iii. strcpy(New->name,q->name);

iv. Set r=r->next and r->next=NULL

e. Move q to the next node, q=q->next

6. return head3, i.e. head of resultant linked list containing union

7. Stop

Flowchart: **Draw Flowchart**

Test cases: **Note: Add input and output here**

Conclusion/analysis:

Thus I have studied concept of set and its representation using array. I have alsoimplemented all the operations of set.

Assignment No.	9 (GROUP D)
Title	Check well formedness of parenthesis
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group D Assignment No 9

Title: Program to check well formedness of parenthesis.

Objectives: Understand the concept and basic operations of Stack in Datastructure.

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.

Outcomes:

Will be able to perform the basic operations of Stack in DataStructure.

Software & Hardware requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC

Theory- Concept in brief:

Stack is abstract data type and linear data structure.

A Stack is data structure in which addition of new element or deletion of existing element always takes place at a same end. This end is known as the top of the stack. That means that it is possible to remove elements from a stack in reverse order from the insertion of elements into the stack.

One other way of describing the stack is as a last in, first out(LIFO) abstractdata type and linear data structure.

Operations on Stack

The stack is basically performed two operations PUSH and POP. Push and pop are the operations that are provided for insertion of an element into the stack and the removal of an element from the stack, respectively.

PUSH:- PUSH operation performed for the adding item to the stack.

POP:- POP operation performed for removing an item from a stack.

ISEMPTY : which returns true if Stack is empty else false; ISFULL : which returns true is Stack is full else false;

Application of Stack:

1. Expression Evolution Expression conversion Infix to Postfix
2. Infix to Prefix Postfix to Infix Prefix to Infix Parsing
3. Simulation of recursion Function call

Algorithms:

Algorithm to check well formedness of parenthesis

1) Declare a character stack S.
2) Now traverse the expression string exp.
   A. If the current character is a starting bracket („(„ or „{„ or „[„) then push it to stack.
   B. If the current character is a closing bracket („)“ or „}“ or „]“) then pop from stack and if the popped character is the matching starting bracket then fine else parenthesis are not balanced.
3) After complete traversal, if there is some starting bracket left in stack then “not balanced”

Algorithm push(char item) Precondition:

Accept a character variable

Post condition: Stack with newly inserted element

Return: Nil

1. if(st.top==(max-1))
   a. Display message "Stack Overflow"
   b. Go to step 4
2. Increment top by 1
3. st.stack[st.top]=item
4. Stop

Algorithm pop()

Precondition: An already created stack

Postcondition : Element at the top of the stack

Return : char

1. if(st.top==-1)
   a. Display message —Stack underflow."
   b. Go to step 3
2. return (st.stack[st.top--])
3. Stop

Flowchart: Draw Flowchart Testcases:

1. (A+(B*C)Expression is invalid

2. (A+(B*C)) Expression is valid

Conclusion/analysis:

Understood the basic concepts of Stack operation Push and Pop and able tocheck whether an expression is well parenthesized.

Assignment No.	10 (GROUP D)
Title	Infix to Postfix Conversion and Evaluation
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group D Assignment No 10

Title: Infix to Postfix Conversion and Evaluation

Objective:

1. Understand the concept of how to convert infix to postfix expression.
2. Understand how to evaluate the expression using stack.

Problem Statement:

Implement C++ program for expression conversion as infix to postfix and itsevaluation using stack based on given conditions:

1. Operands and operator, both must be single character.
2. Input Postfix expression must be in a desired format.
3. Only '+', '-', '*' and '/ ' operators are expected.

Outcome:

1. Will be able to understand the concept of how to convert infix topostfix expression.
2. Will be able to understand how to evaluate the expressionusing stack.

Software & Hardware Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC

Theory:

Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.

Postfix expression: The expression of the form a b op. When an operator isfollowed for every pair of operands.

Need of postfix representation of the expression

The compiler scans the expression either from left to right or from right to left.Consider the below expression: a op1 b op2 c op3 d If op1 =

+, op2 = *, op3 = +

The compiler first scans the expression to evaluate the expression b * c, thenagain scan the expression to add a to it. The result is then added to d after another scan. The repeated scanning makes it very in- efficient. It is better to convert the expression to postfix (or prefix) form before evaluation. The corresponding expression in postfix form is: abc*+d+. The postfix expressions can be evaluated easily using a stack To implement conversion of an infix

expression into postfix expression.

Algorithm:

1. Define a stack of to hold the characters.
2. Initialize stack top = -1.
3. Read the infix expression.
4. Add „)“ to the end of the infix expression.
5. Push „(„to the stack.
6. Scan the infix expression from left to right and repeat the step7 forall the characters in the infix expression.
7. If the character is an operand Add it to the postfix expression If the character is „(„ Push it to the stack If the character is „)“ Repeatedly Pop the characters from the stack and add it to the postfix expression until „)“ is encountered. Pop „)“ If the character isan operator If the precedence of the character is lesser than orequal to the precedence of the operator in the top of the stack repeatedly pop the characters from the stack and add it to the post fix expression till an operator of higher precedence is encountered. Push the operator to the top of the stack.
8. Print the postfix expression.
9. Stop.

Conversion To Postfix

EXAMPLE:

A+(B*C-(D/E-F)*G)*H

To implement evaluation of a postfix expression.Algorithm for Postfix Evaluation:

Stack	Input	Output
Empty	A+(B*C-(D/E-F)*G)*H	-
Empty	+(B*C-(D/E-F)*G)*H	A
+	(B*C-(D/E-F)*G)*H	A
+(	B*C-(D/E-F)*G)*H	A
+(	*C-(D/E-F)*G)*H	AB
+(*	C-(D/E-F)*G)*H	AB
+(*	-(D/E-F)*G)*H	ABC
+(-	(D/E-F)*G)*H	ABC*
+(-(	D/E-F)*G)*H	ABC*
+(-(	/E-F)*G)*H	ABC*D
+(-(/	E-F)*G)*H	ABC*D
+(-(/	-F)*G)*H	ABC*DE
+(-(-	F)*G)*H	ABC*DE/
+(-(-	F)*G)*H	ABC*DE/
+(-(-	)*G)*H	ABC*DE/F
+(-	*G)*H	ABC*DE/F-
+(-*	G)*H	ABC*DE/F-
+(-*	)*H	ABC*DE/F-G
+	*H	ABC*DE/F-G*-
+*	H	ABC*DE/F-G*-
+*	End	ABC*DE/F-G*-H
Empty	End	**ABC*DE/F-G*-H*+**

1. Declare the structure for the stack.
2. Read the postfix expression.
3. Repeatedly execute the following for all the characters in theexpression from left to right If the character is a number, then convert the character to integer by subtracting „0“ from the character. Push the integer value into the stack. If the character is a operator Pop two values from the stack and perform the operation and store the result to stack.
4. The stack top has the result. Pop it and Print.
5. Stop

Flowchart:

Draw Flowchart

Test Cases:

Input:

Infix Expression: (3*2) / (5-3)

Output:

Postfix Expression: 3 2 * 5 3 - /

Evaluation result: 3

Conclusion/Analysis:

Understood the concept of how to convert infix to postfix expression and how toevaluate the expression using stack**.**

Assignment No.	11 (GROUP E)
Title	Perform different operations on Queue
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group E- Assignment no: 11

Title: Perform different operations on Queue.

Objectives:

1. Understand the basic concept of Queue Implementation
2. Understand the different operations that can be performed on queue such as addition and deletion

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

Outcomes: Upon completion of assignment, student will be able to:

1. Understand the concept of Queue operations
2. Perform addition and deletion of job operations on queue.

Software Requirements: 64-bit Open source Linux or its derivative, Windows8/9/10,Programming Tools: G++/GCC compiler, Eclipse IDE.

Hardware Requirements: Intel ® Core™ i3-3220 CPU @3.30 GHz, 4 GB RAM, 500 GBATA HDD

Theory Concept in Brief:

Queue:

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is openat both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

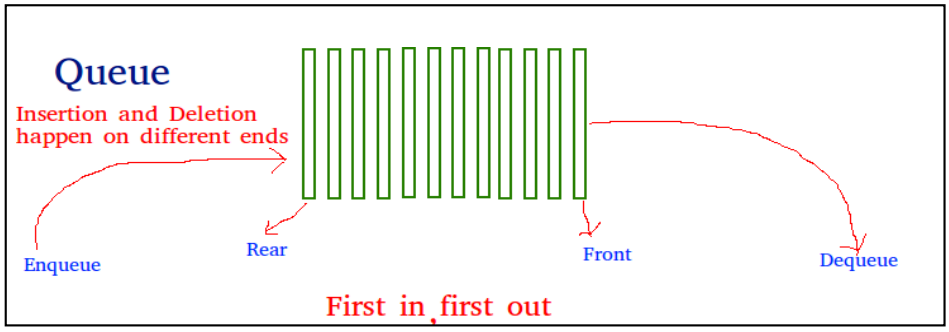


Fig. Queue Representation

Basic Operations of Queue:

A queue is an object or more specifically an abstract data structure (ADT) that allows thefollowing operations:

1. **Enqueue:** Add an element to the end of the queue
2. **Dequeue:** Remove an element from the front of the queue
3. **IsEmpty:** Check if the queue is empty
4. **IsFull:** Check if the queue is full
5. **Peek:** Get the value of the front of the queue without removing it.

Applications of Queue Data Structure:

1. CPU scheduling, Disk Scheduling
2. When data is transferred asynchronously between two processes. The queue is usedfor synchronization.eg: IO Buffers, pipes, file IO, etc.
3. Handling of interrupts in real-time systems.
4. Call Center phone systems use Queues to hold people calling them in an order

Types of Queue:

Simple Queue:

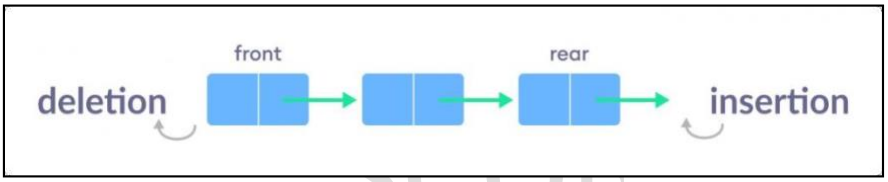


Fig. Single Queue Representation

In a simple queue, insertion takes place at the rear and removal occurs at the front. It strictlyfollows FIFO rule.

Circular Queue:

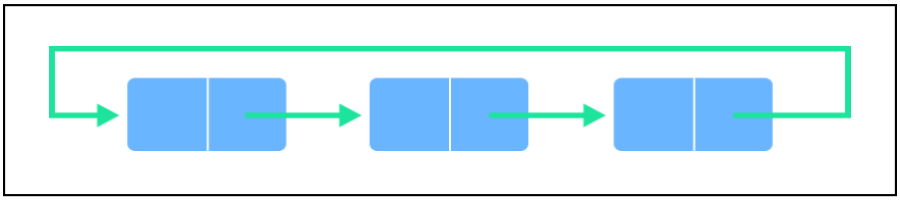


Fig. Circular Queue Representation

In a circular queue, the last element points to the first element making a circular link.

Priority Queue:

A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue. Insertion occurs based on the arrival of the values and removal occurse based on priority.

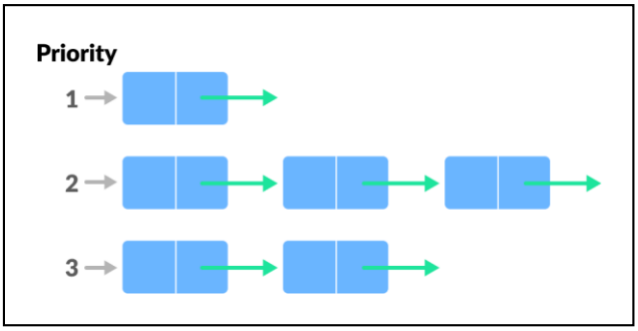


Fig. Circular Queue Representation

Deque (Double Ended Queue):

Double Ended Queue is a type of queue in which insertion and removal of elements can be performed from either from the front or rear. Thus, it does not follow FIFO rule (First In First Out).

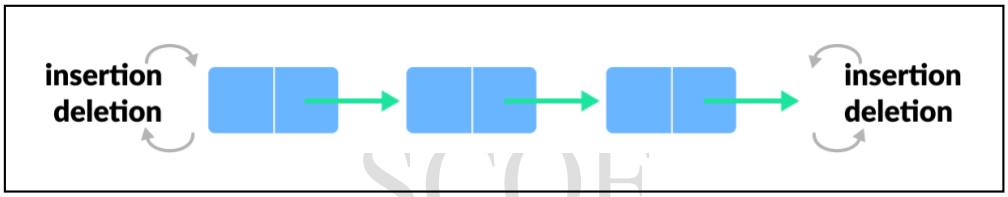


Fig. Deque Representation

Working of Queue:

Queue operations work as follows: two pointers FRONT and REAR

FRONT track the first element of the queue, REAR tracks the last elements of the queue.Initially, set value of FRONT and REAR to -1

Enqueue Operation:

Queues maintain two data pointers, front and rear. Therefore, its operations are comparatively difficult to implement than that of stacks.

Algorithm:

The following steps should be taken to enqueue (insert) data into a queue −

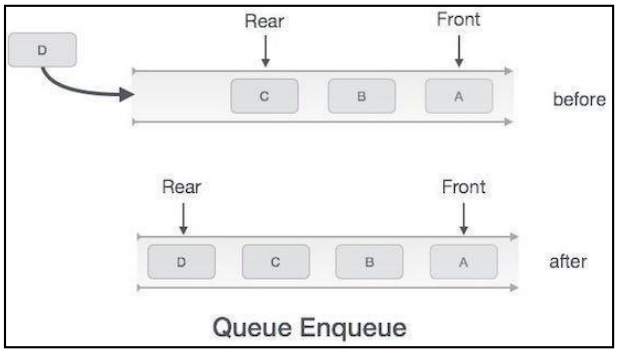
Step 1 − Check if the queue is full.

Step 2 − If the queue is full, produce overflow error and exit.

Step 3 − If the queue is not full, increment rear pointer to point the next empty space.

Step 4 − Add data element to the queue location, where the rear is pointing.

Step 5 − return success.



Dequeue Operation:

Accessing data from the queue is a process of two tasks − access the data where **front** is pointing and remove the data after access.

Algorithm:

The following steps are taken to perform **dequeue** operation −

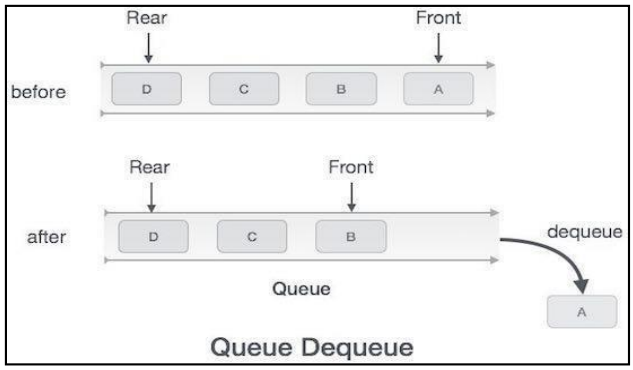
Step 1 − Check if the queue is empty.

Step 2 − If the queue is empty, produce underflow error and exit.

Step 3 − If the queue is not empty, access the data where**front** is pointing.

Step 4 − Increment **front** pointer to point to the next available data element.

Step 5 − Return success.



Complexity Analysis:

The complexity of enqueue and dequeue operations in a queue using an array is O(1).

Algorithm of isfull() function −

begin procedure isfull

if rear equals to MAXSIZEreturn true

else return false

end if

end procedure

Algorithm of isempty() function –

begin procedure isempty

if front is less than MIN OR front is greater than rear

return true

else return false

end if

end procedure

Flowchart:

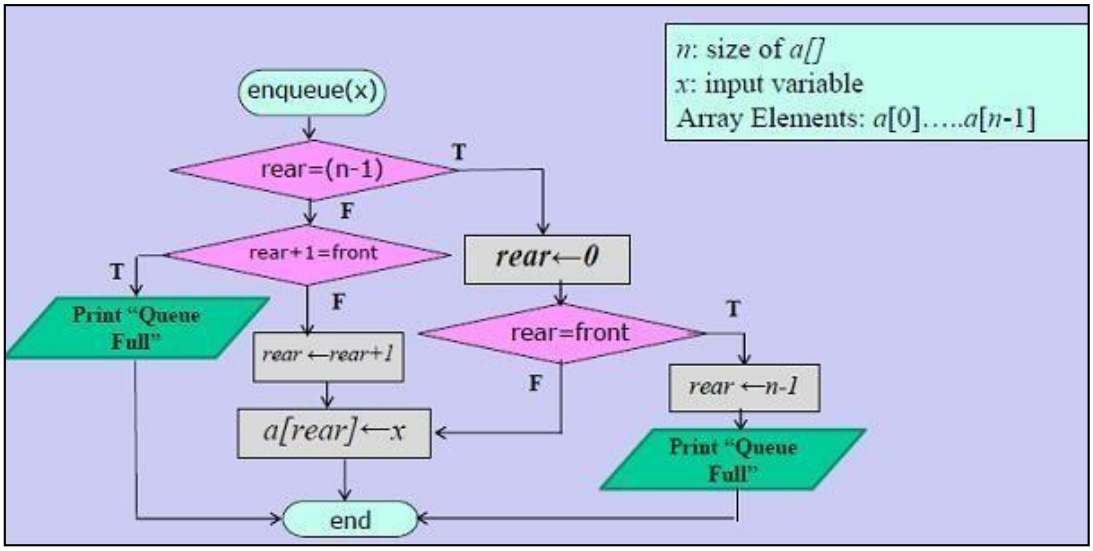


Fig. Flowchart for Enqueue() function

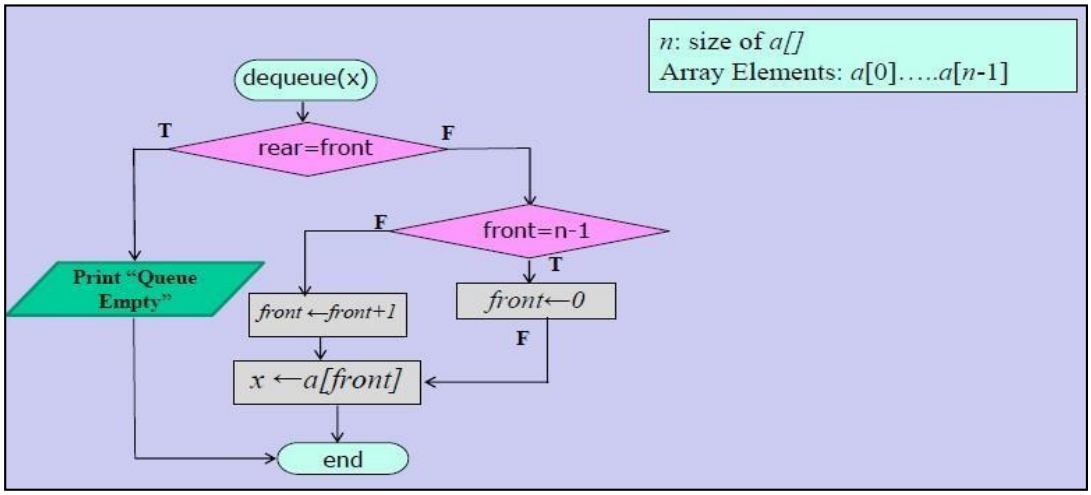


Fig. Flowchart for Dequeue() function

Test Cases:

Test case number: 1

* Test case values: push 1 and 2 into the queue
* Expected output (Post-state): [1, 2]

Test case number: 2

* Test case values: push 1, 2 and 3 into a queue with 2 spaces
* Expected output (Post-state): throw Illegal State Exception

Test case number: 3

* Test case values: push 1 into a queue with 2 spaces, check if the queue is full
* Expected output (Post-state): return false, the queue is not full

Conclusion: By this way, we understood the concept of queue and its different operations.

Question Bank:

1. What is Queue?
2. What are the different operations that can be performed on queue?
3. Explain all the operations on queue
4. Which are different types of queues , Explain.

Assignment No.	12 (GROUP E)
Title	Perform operations on Double ended queue
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group E- Assignment no: 12

Title: Perform operations on Double ended queue

Objectives:

1. Understand the basic concept of Double-ended Queue Implementation.
2. To simulate deque with functions to add and delete elements from either end of thedeque.

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.

Outcomes: Upon completion of assignment, student will be able to:

Will be able to perform insertion and deletion in double ended queue.

Software Requirements: 64-bit Open source Linux or its derivative, Windows8/9/10,Programming Tools: G++/GCC compiler, Eclipse IDE.

Hardware Requirements: Intel ® Core™ i3-3220 CPU @3.30 GHz, 4 GB RAM, 500 GBATA HDD

Theory Concept in Brief:

De Queue:

Double-ended queue is an abstract data type similar to an simple queue, it allows you to insert and delete from both sides means items can be added or deleted from the front or rear end. It is also often called a **head-tail linked list**, also refers to a specific data structure implementation of a deque.

The DeQueue is represented as follows.

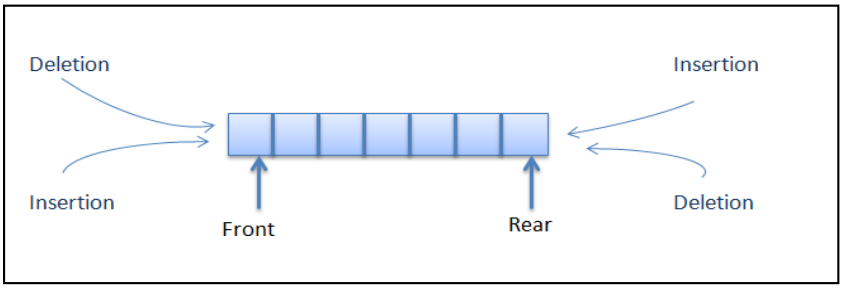


Fig. DeQueue Representation

Mainly the following four basic operations are performed on queue:

insertFront(): Adds an item at the front of Deque. ***insertLast()***: Adds an item at the rear of Deque. ***deleteFront()***: Deletes an item from front of Deque. ***deleteLast()***: Deletes an item from rear of Deque.

In addition to above operations, following operations are also supported

getFront(): Gets the front item from queue. ***getRear()***: Gets the last item from queue. ***isEmpty()***: Checks

whether Deque is empty or not.***isFull()***: Checks whether Deque is full or not.

Types of De Queue

1. Input Restricted DeQueue
2. Output Restricted DeQueue

In Input Restricted DeQueue , insertion can be done from REAR only, but deletion can be done from both FRONTand REAR.

In Output Restricted DeQueue,deletion can be done from FRONT only, but insertion can be donefrom both FRONT and REAR.

Applications of Deque Data Structure

In undo operations on software.

- To store history in browsers.
- For implementing both stacks and queues.

Algorithm for Insertion at rear end

Step -1: [Check for overflow]

if(rear==MAX)

print("Queue is Overflow");

return;

Step-2: [Insert element]

else rear=rear+1;

q[rear]=no;

[Set rear and front pointer]

if rear=0 rear=1;

if front=0front=1;

Step-3: return

Algorithm for Insertion at front end

Step-1: [Check for the front position]

if(front<=1)

Print ("Cannot add item at front end");

return;

Step-2: [Insert at front]

else front=front-1;

q[front]=no;

Step-3: return

Algorithm for Deletion from front end Step-1 [Check for front pointer]

if front=0

print(" Queue is Underflow");

return;

Step-2 [Perform deletion]

else no=q[front];

print("Deleted element is",no);

[Set front and rear pointer]

if front=rear front=0;

rear=0;

else front=front+1;

Step-3 : Return

Algorithm for Deletion from rear end

Step-1 : [Check for the rear pointer]

if rear=0

print("Cannot delete value at rear end");

return;

Step-2: [perform deletion]

else no=q[rear];

[Check for the front and rear pointer]

if front= rear front=0;

rear=0;

else

rear=rear-1;

print("Deleted element is",no);

Step-3 : Return

Flowchart:

Draw flowchart for above algorithms

Test Cases:

Enter Integer Data :2

1: Insert From Left

2: Insert From Right

3: Delete From left

4: Delete FromRight

5: Display

6: Exit Program

Enter Your Choice:2

Enter Integer Data :5

1: Insert From Left

2: Insert From Right

3: Delete From left

4: Delete FromRight

5: Display

6: Exit Program

Conclusion:

By this way, we understood the concept of simulation of deque with functions to add anddelete elements from either end of the deque.

Question Bank:

1. What is queue?Types of queue
2. What is double ended queue?
3. How to insert the new node in Doubly Link List?
4. How to delete the node from front of Doubly Link List ?How to delete the node from end of Doubly Link List ?How to delete the node in between of Doubly Link List

Assignment No.	13 (GROUP E)
Title	Perform the different operations on Circular Queue
Subject	Data Structures Laboratory
Class	S.E. (C.E.)
Roll No.	
Date	
Signature	

Group E- Assignment no: 13

Title: Perform the different operations on Circular Queue.

Objectives:

1. Understand the basic concept of Circular Queue Implementation.
2. Understand the concept of insertion, deletion in a circular 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.

Outcomes: Upon completion of assignment, student will be able to:

- Will be able to understand the concept of insertion and deletion in circular queue**Software**

Requirements: 64-bit Open source Linux or its derivative, Windows 8/9/10,Programming Tools: G++/GCC compiler, Eclipse IDE.

Hardware Requirements: Intel ® Core™ i3-3220 CPU @3.30 GHz, 4 GB RAM, 500 GBATA HDD

Theory Concept in Brief:

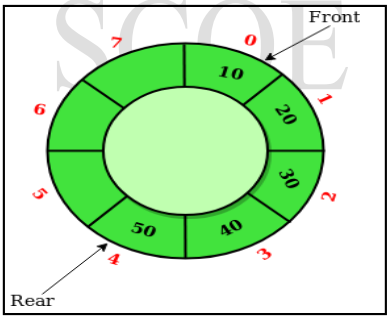


Fig. Circular Queue Representation

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'**.

Basic Operations

Some of the basic operations of the circular queue are as follows:

1. **Front:** Returns the front position in the circular queue.
2. **Rear:** Returns the rear position in the circular queue.
3. **Enqueue:** Enqueue (value) is used to insert an element in the circular queue. The elementis always inserted at the rear end of the queue.

   We follow the following sequence of steps to insert a new element in the circular queue.

- Check if the circular queue is full: test ((rear == SIZE-1 && front == 0) || (rear == front- 1)), where SIZE‟is the size of the circular queue.
- If the circular queue is full then it displays a message as "Queue is full". If queue is not full then, check if (rear == SIZE – 1 && front != 0). If it is true then set rear=0 and insert element.

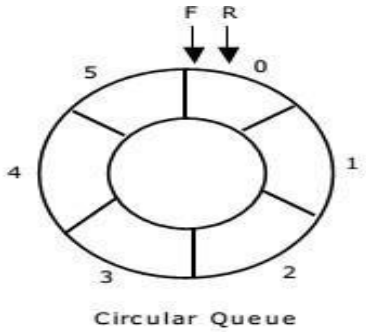
4. **Dequeue:** Dequeue function is used to delete an element from the queue. In the circular queue, the element is always deleted from the front end. Given below is the sequence for dequeue operation in a circular queue.

Steps:

1) Check if the circular queue is Empty: check if (front==-1).
2) If it is empty then display the message "Queue is empty". If queue is not empty then performstep 3.
3) Check if (front==rear). If it is true then set front=rear= -1 else check if (front==size-1), if it is true then set front=0 and return the element.

Application of Circular Queue:

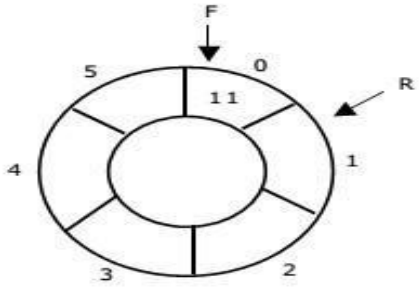
1. **CPU Scheduling:** Operating system process that requires some event to occur or for some other processes to complete for execution is often maintained in a circular queue so that they execute one after the other when all the conditions are met or when all events occur.
2. **Memory Management:** Use of ordinary queues wastes memory space as already mentioned in our above discussion. Using a circular queue for memory management is beneficial for optimum memory usage.
3. **Computer Controlled Traffic Signal System:** Computerized traffic signals are often added to a circular queue so that they repeat themselves after the specified time interval has elapsed.



Circular Queue

Queue Empty
MAX = 6
FRONT = REAR = 0
COUNT = 0

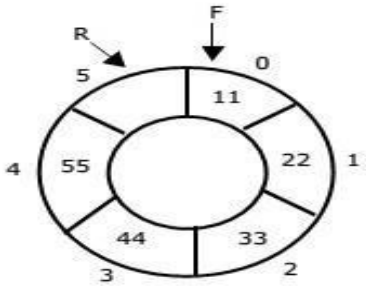
Now, insert 11 to the circular queue. Then circular queue status will be:



Circular Queue

FRONT = 0
REAR = (REAR + 1) % 6 = 1
COUNT = 1

Insert new elements 22, 33, 44 and 55 into the circular queue. The circular queue status is:



Circular Queue

FRONT = 0
REAR = (REAR + 1) % 6 = 5
COUNT = 5

Fig. Representation of Circular Queue:

Let us consider a circular queue, which can holdmaximum (MAX) of six elements. Initially queue is empty.

Drawback of Circular Queue

The drawback of circular queue is , difficult to distinguished the full and empty cases. It is alsoknown as

Boundary case problem.

1. In circular queue it is necessary that:
2. Before insertion, fullness of Queue must be checked (for overflow).
3. Before deletion, emptiness of Queue must be checked (for underflow).

Circular Queue Complexity Analysis:

The complexity of the enqueue and dequeue operations of a circular queue is O(1) for (array implementations).

Algorithm of a Circular queue:

1. Initialize the queue, with size of the queue defined (maxSize), and head and tail pointers.
2. enqueue: Check if the number of elements is equal to maxSize - 1: If Yes, then return Queue is full.

   If No, then add the new data element to the location of tail pointer and increment the tail pointer.
3. dequeue: Check if the number of elements in the queue is zero: If Yes, then return Queue is empty.

   If No, then increment the head pointer.
4. Finding the size:

   If, tail >= head, size = (tail - head) + 1

   But if, head > tail, then size = maxSize - (head - tail) + 1

Flowchart:

Draw Flowchart for above operations.

Conclusion/Analysis

In this way, we understood the concept of insertion and deletion in circular queue

Questions:

1. What is circular queue?
2. What are the basic operations of Circular Queue?
3. Explain the applications of Circular Queue.
4. Difference between Circular Queue and Priority Queue.
5. What is Complexity Analysis of Circular Queue.