



Empowerment Through Quality Technical Education
AJEENKYA DY PATIL SCHOOL OF ENGINEERING

Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105

Website: <https://dypsoe.in/>

LAB MANUAL

Data Structures Laboratory (217522)

SE (AI&DS) 2020 COURSE

Course Coordinator

Prof. Rohini Shrikhande

**DEPARTMENT OF
ARTIFICIAL INTELLIGENCE & DATA SCIENCE**

Department of Artificial Intelligence & Data Science

Vision:

Imparting quality education in the field of Artificial Intelligence and Data Science

Mission:

- To include the culture of R and D to meet the future challenges in AI and DS.
- To develop technical skills among students for building intelligent systems to solve problems.
- To develop entrepreneurship skills in various areas among the students.
- To include moral, social and ethical values to make students best citizens of country.

Program Educational Outcomes:

1. To prepare globally competent graduates having strong fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.
2. To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.
3. To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.
4. To prepare the graduates with strong managerial and communication skills to work effectively as individuals as well as in teams.

Program Specific Outcomes:

1. Professional Skills- The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.

2. Problem-Solving Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

3. Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Table of Contents

Contents

| | |
|---------------------------------------|-----|
| 1. Guidelines to manual usage | 4 |
| 2. Laboratory Objective | 8 |
| 3. Laboratory Equipment/Software..... | 8 |
| 4. Laboratory Experiment list | 9 |
| 1 Experiment No. 1 | 11 |
| 2 Experiment No. 2 | 15 |
| 3 Experiment No. 3 | 19 |
| 4 Experiment No. 4 | 19 |
| 5 Experiment No. 5 | 212 |
| 6 Experiment No. 6 | 245 |
| 7 Experiment No. 7 | 267 |
| 8 Experiment No. 8 | 301 |
| 9 Experiment No. 9 | 304 |
| 10 Experiment No. 10 | 307 |
| 11 Experiment No. 11 | 41 |
| 12 Experiment No. 12 | 45 |
| 13 Experiment No. 13 | 48 |

1. Guidelines to manual usage

This manual assumes that the facilitators are aware of collaborative learning methodologies.

This manual will provide a tool to facilitate the session on Digital Communication modules in collaborative learning environment.

The facilitator is expected to refer this manual before the session.

Icon of Graduate Attributes

| | | | |
|---------------------------------------|--------------------------------------|--|--|
| K Applying Knowledge | A Problem Analysis | D Design & Development | I Investigation of problems |
| M Modern Tool Usage | E Engineer & Society | E Environment Sustainability | T Ethics |
| T Individual & Team work | O Communication | M Project Management & Finance | I Life-Long Learning |

Disk Approach- Digital Blooms Taxonomy



- 1: Remembering / Knowledge**
- 2: Comprehension / Understanding**
- 3: Applying**
- 4: Analyzing**
- 5: Evaluating**
- 6: Creating / Design**

Program Outcomes:

1. **Engineering knowledge:** An ability to apply knowledge of mathematics, including discrete mathematics, statistics, science, computer science and engineering fundamentals to model the software application.
2. **Problem analysis:** An ability to design and conduct an experiment as well as interpret data, analyze complex algorithms, to produce meaningful conclusions and recommendations.
3. **Design/development of solutions:** An ability to design and development of software system, component, or process to meet desired needs, within realistic constraints such as economic, environmental, social, political, health & safety, manufacturability, and sustainability.
4. **Conduct investigations of complex problems:**An ability to use research based knowledge including analysis, design and development of algorithms for the solution of complex problems interpretation of data and synthesis of information to provide valid conclusion.
5. **Modern tool usage:** An ability to adapt current technologies and use modern IT tools, to design, formulate, implement and evaluate computer based system, process, by considering the computing needs, limits and constraints.
6. **The engineer and society:** An ability of reasoning about the contextual knowledge of the societal, health, safety, legal and cultural issues, consequent responsibilities relevant to IT practices.
7. **Environment and sustainability:** An ability to understand the impact of engineering solutions in a societal context and demonstrate knowledge of and the need for sustainable development.
8. **Ethics:** An ability to understand and commit to professional ethics and responsibilities and norms of IT practice.
9. **Individual and team work :**An ability to apply managerial skills by working effectively as an individual, as a member of a team, or as a leader of a team in multidisciplinary projects.
10. **Communication:** An ability to communicate effectively technical information in speech, presentation, and in written form
11. **Project management and finance:** An ability to apply the knowledge of Information Technology and management principles and techniques to estimate time and resources needed to complete engineering project.
12. **Life-long learning:** An ability to recognize the need for, and have the ability to engage in independent and life-long learning.

Course Name: Data Structures Laboratory

Course Code: (217522)

Course Outcomes

1. **CO1:** Use algorithms on various linear data structure using sequential organization to solve real life problems.
2. **CO2:** Analyze problems to **apply** suitable searching and sorting algorithm to various applications.
3. **CO3:** Analyze problems to **use variants of** linked list and solve various real life problems.
4. **CO4:** Designing and implement data structures and algorithms for solving different kinds of problems

CO to PO Mapping:

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|
| CO1 | 1 | 1 | 2 | 1 | - | - | - | - | - | - | - | - |
| CO2 | 2 | 2 | 2 | 1 | - | - | - | - | - | - | - | - |
| CO3 | - | 2 | 1 | 1 | - | - | - | - | - | - | - | - |
| CO4 | 1 | 2 | 2 | 1 | - | - | - | - | - | - | - | - |

CO to PSO Mapping:

| | PSO1 | PSO2 | PSO3 |
|------------|------|------|------|
| CO1 | | | |
| CO2 | | | |
| CO3 | | | |
| CO4 | | | |

2. Laboratory Objective

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

3. Laboratory Equipment/Software

Operating System recommended:- 64-bit Open source Linux or its derivative **Programming tools recommended:-** - Open Source Python, Programming tool like Jupyter Notebook, Pycharm, Spyder, G++/GCC

4. Laboratory Experiment list

| Sr. No. | Title |
|---------|---|
| | List of Assignments |
| 1 | <p>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: -</p> <ul style="list-style-type: none">a) List of students who play both cricket and badmintonb) List of students who play either cricket or badminton but not bothc) Number of students who play neither cricket nor badmintond) Number of students who play cricket and football but not badminton.e) (Note- While realizing the group, duplicate entries should be avoided, Do not use SET built-in functions) |
| 2 | <p>Write a Python program to store marks scored in subject "Fundamental of Data Structure" by N students in the class. Write functions to compute following:</p> <ul style="list-style-type: none">a) The average score of classb) Highest score and lowest score of classc) Count of students who were absent for the testd) Display mark with highest frequency |
| 3 | <p>e) Write a Python program for department library which has N books, write functions for following:</p> <ul style="list-style-type: none">a) Delete the duplicate entriesb) Display books in ascending order based on cost of booksc) Count number of books with cost more than 500.d) Copy books in a new list which has cost less than 500. |
| 4 | <p>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</p> <ul style="list-style-type: none">a) Selection Sortb) Bubble sort and display top five scores. |
| 5 | <p>Write a Python program to store second year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using :</p> <ul style="list-style-type: none">a) Insertion sortb) Shell Sort and display top five scores. |
| 6 | <p>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.</p> |

| | |
|--------------------------------|--|
| 7 | <p>Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and 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:</p> <ol style="list-style-type: none"> 1. Add and delete the members as well as president or even secretary. 2. Compute total number of members of club 3. Display members 4. Two linked lists exist for two divisions. Concatenate two lists. |
| 8 | <p>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 :</p> <p>Write C/C++ program to store two sets using linked list, compute and display:</p> <ol style="list-style-type: none"> a. Set of students who like either vanilla or butterscotch or both b. Set of students who like both vanilla and butterscotch c. Number of students who like neither vanilla nor butterscotch |
| 9 | <p>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-</p> <ol style="list-style-type: none"> 1. To print original string followed by reversed string using stack 2. To check whether given string is palindrome or not |
| 10 | <p>Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions:</p> <ol style="list-style-type: none"> 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. |
| 11 | <p>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.</p> |
| 12 | <p>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.</p> |
| 13 | <p>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.</p> |
| Content Beyond Syllabus | |
| 1 | Implement a real time mini project using ADT |

Experiment No. 1

Aim:

Perform various set operations in python

Objective:

- To understand the concept of array.
- To understand various set operation using array and function.
- To understand use of array data structure for set 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: -

- List of students who play both cricket and badminton
- List of students who play either cricket or badminton but not both
- Number of students who play neither cricket nor badminton
- 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-in functions)

Theory:

Set:

Definition: A set is any collection of definite, distinguishable objects, and we call these objects the elements of the set.

To represent a set, array data structure is used.

Array:

Array is a data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations.

In single dimensional array we can represent all set which is required to perform various set operations.

Operations on set:

Definition: Let A and B be subsets of a set U.

1. Union of A and B: $A \cup B = \{x \in U \mid x \in A \text{ or } x \in B\}$

Union of the sets A and B, denoted $A \cup B$, is the set of all objects that are a member of A, or B, or both. The union of $\{1, 2, 3\}$ and $\{2, 3, 4\}$ is the set $\{1, 2, 3, 4\}$. We can represent the above set in terms of programming using array data structure as follows:

$A[] = \{1, 2, 3\}$ and $B[] = \{2, 3, 4\}$

Suppose C is set for storing union result.

So therefore union is $C[] = \{1, 2, 3, 4\}$.

2. Intersection of A and B: $A \cap B = \{x \in U \mid x \in A \text{ and } x \in B\}$

Intersection of the sets A and B, denoted $A \cap B$, is the set of all objects that are members of both A and B. The intersection of $\{1, 2, 3\}$ and $\{2, 3, 4\}$ is the set $\{2, 3\}$.

Intersection using array:

$A[] = \{1, 2, 3\}$ and $B[] = \{2, 3, 4\}$

Suppose C is set for intersection union result.

So therefore intersection is $C[] = \{2, 3\}$.

3. Difference of B minus A:

$B - A = \{x \in U \mid x \in B \text{ and } x \notin A\}$

The difference of two sets B and A is the collection of objects in B that are not in A.

The difference is written $B - A$.

4. Complement of A: $A^c = \{x \in U \mid x \notin A\}$

The complement of a set A is the collection of objects in the universal set that are not in A.

Algorithm:

Algorithm for union of two set:

Algorithm Union(arr1[], arr2[])

For union of two arrays, follow the following procedure.

(Write your own algorithm/pseudocode)

Algorithm for Intersection of two set:

Algorithm Intersection(arr1[], arr2[])

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

(Write your own algorithm/pseudocode)

Algorithm for to find the difference of two set:

(Write your own algorithm/pseudocode)

Algorithm to find out complement of two set:

(Write your own algorithm/pseudocode)

Input:

Output:

Various operation of set is tested.

Union operation :

Set A : student who are playing cricket

11 12 13 15

Set B: student who are playing badminton

15 16 17 18

Union set:

11 12 13 15 16 17 18.

Conclusion:

Thus we have implemented pre-processing of a text document such as stop word removal, stemming.

Outcome:

Upon completion of this experiment, students will be able to:

- Understand the concept of array.
- Understand the use of functions for set operations.
- Understand the use of array data structure for set operations

Experiment No. 2

Aim:

Perform different operations on array in python.

Objective:

- To understand the concept of data structure
- To understand linear and static data structure
- To understand array data structure
- To understand array as an ADT

Problem Statement:

Write a Python program to store marks scored in subject “Fundamental of Data Structure” by N students in the class. Write functions to compute following:

- a) The average score of class
- b) Highest score and lowest score of class
- c) Count of students who were absent for the test
- d) Display mark with highest frequency

Software Requirements:

Open Source Python, Programming tool like Jupyter Notebook, Pycharm, Spyder (64 bit)

Theory:

Data structure is:

- ☐ A representation of data and the operations allowed on that data or
- ☐ A way in which sets of data are organized in a particular system or
- ☐ A computer interpretable format used for storing, accessing, transferring, and archiving data.

Data structures are categorized into two types

(a) Linear : Elements in linear data structure form a sequence. A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached.

Eg: Arrays, Stacks, Queues, Linked Lists.

(b) Non-Linear : If the elements of data structure do not form a sequence or a linear list then data structure is called as Non-Linear data structure. eg. Trees , Graphs.

ARRAY:

Array is defined as a collection of elements of same data type.

Hence, it is also called as homogeneous collection of data items.

An array:

- is an indexed sequence of components.
- component has a fixed, unique index.
- typically, occupies sequential storage locations.
- length is determined when the array is created, and cannot be changed (static data structure).
- has direct access to each element by specifying its position.

Position of element in an array defines index or subscript of particular element. Array index always starts with zero. If the size of array is n then index ranges from 0 to n-1.

Array index must be integer or integer expression. There can be array of integer, floating point or character values. Character array is called as String.

Syntax: type arrayName [arraySize];

Index 0 1 2 3 4

Value 60 20 85 35 75

e.g.

- `int data[5]; data[0]= 60; data[1]=20; data[2]=85; data[3]=35; data[4]=75; or`
- `int data[5]={60, 20, 85, 35, 75}; or`
- `int data[] = {60, 20, 85, 35, 75};`

in above example array is declared and initialized too.

Array initialization can only be used in a declaration.

An array is an Abstract Data Type

The array type has a set of operations that can be applied uniformly to each of these values

- The only operation is indexing
- Alternatively, this can be viewed as two operations:
 - inspecting an indexed location
 - storing into an indexed location

It's abstract because the implementation is hidden: all access is via the defined operations

Algorithm:

Algorithm to accept n student's marks:

1. Start
2. Read the marks of n students and stored in array (-1 for absent student)
3. Calculate and print the average of n student's marks
4. Comparing all student's marks print highest marks
5. Comparing all student's marks print lowest marks
6. Comparing all student's marks print mark scored by most of the students
7. Comparing -1 in an array, print absent students
8. Stop

Algorithm for highest mark in an array:

(Write your own algorithm/pseudocode)

Algorithm for lowest mark in an array:

(Write your own algorithm/pseudocode)

Algorithm for mark scored by most of the students:

(Write your own algorithm/pseudocode)

Algorithm for calculating absent students:
(Write your own algorithm/pseudocode)

Input:

Output:

Conclusion:

Thus, we implemented linear-static data structure (array) by storing marks of students in an array.

Outcomes:

- Understanding the need of data structure
- Understanding the static data structure, array.
- Understanding the linear data structure, array.

Experiment No. 3

Aim:

Write a Python program to store marks scored in subject “Fundamental of Data Structure”. Write functions to compute following: Write a Python program for department library which has N books, write functions for following:

- a) Delete the duplicate entries
- b) Display books in ascending order based on cost of books
- c) Count number of books with cost more than 500.
- d) Copy books in a new list which has cost less than 500.

Objective:

To study-

- To understand the concept of data structure
- To implement suitable data structures
- To perform basic operations on data structure

Theory:

(Write your own algorithm/pseudocode)

Input:

Output:

Conclusion:

Thus, we implemented a library application and performed various operation

Outcomes:

- Understanding the need of data structure
- Apply suitable data structures
- Perform various operations on it

Experiment No. 4

Aim:

Implementation of Selection sort and bubble sort in python

Objectives:

- a) To understand the concept of sorting.
- b) To understand the concept and use of selection sort.
- c) To understand the concept and use of bubble sort.

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

- a) Selection Sort
- b) Bubble sort and display top five scores.

Theory :

Selection Sort:

Selection sort carries out a sequence of passes over the table. At the first pass an entry is selected on some criteria and placed in the correct position in the table. The possible criteria for selecting an element are to pick the smallest or pick the largest. If the smallest is chosen then, for sorting in ascending order, the correct position to put it is at the beginning of the table.

Now that the correct entry is in the first place in the table the process is repeated on the remaining entries. Once this has been repeated $n-1$ times the $n-1$ smallest entries are in the first $n-1$ places which leaves the largest element in the last place. Thus only $n-1$ passes are required.

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Bubble Sort:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent

elements if they are in wrong order.

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Algorithm:

Input :Unsorted Array: [65 35 45 15 85 5 95 25]

Output :Sorted Array: [5 15 25 35 45 65 85 95]

Conclusion : Thus, we implemented selection sort and bubble sort to sort the given array in ascending order.

Experiment No. 5

Aim:

Implementation of insertion sort and shell sort in python

Objectives:

- a) To understand the insertion sort.
- b) To understand the shell sort.

Problem Statement:

Write a Python program to store second year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using :

- a) Insertion sort
- b) Shell Sort and display top five scores.

Theory :

Insertion Sort:

This is a in-place comparison based sorting algorithm. Here, a sub-list is maintained which is always sorted. For

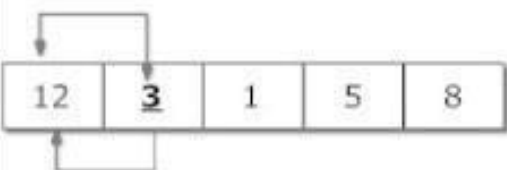
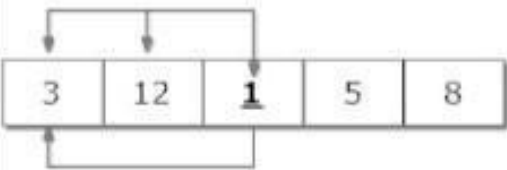
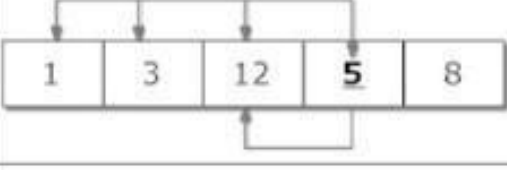
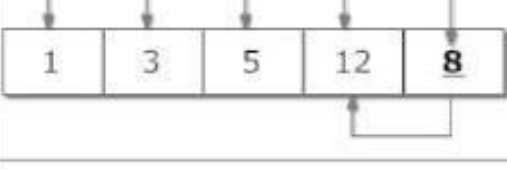

example, the lower part of an array is maintained to be sorted. A element which is to be inserted' in this sorted sub-

list, has to find its appropriate place and insert it there. Hence the name insertion sort.

The array is searched sequentially and unsorted items are moved and inserted into sorted sub-list (in the same array).

This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where n are no.

of items.

| | | |
|---------------|---|--|
| Step 1 |  | Checking second element of array with element before it and inserting it in proper position. In this case, 3 is inserted in position of 12. |
| Step 2 |  | Checking third element of array with elements before it and inserting it in proper position. In this case, 1 is inserted in position of 3. |
| Step 3 |  | Checking fourth element of array with elements before it and inserting it in proper position. In this case, 5 is inserted in position of 12. |
| Step 4 |  | Checking fifth element of array with elements before it and inserting it in proper position. In this case, 8 is inserted in position of 12. |
| |  | Sorted Array in Ascending Order |

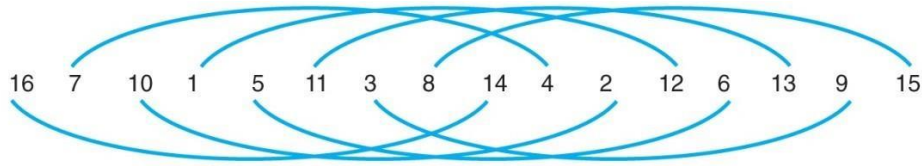
The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

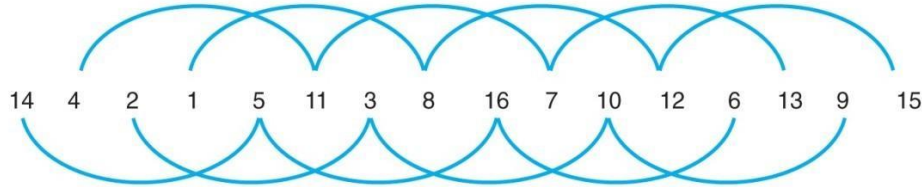
Shell Sort

The shell sort, improves on the insertion sort by breaking the original list into a number of smaller sublists, each of which is sorted using an insertion sort. The unique way that these sublists are chosen is the key to the shell sort. Instead of breaking the list into sublists of contiguous items, the shell sort uses an increment i , sometimes called the gap, to create a sublist by choosing all items that are i items apart.

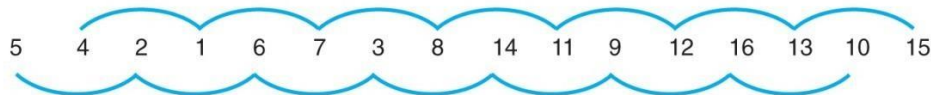
The running time of Shellsort is heavily dependent on the gap sequence it uses. For many practical variants, determining their time complexity remains an open problem.



(a) Pass 1



(b) Pass 2



(c) Pass 3



(d) Pass 4

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Algorithm:

Input :Unsorted Array: [65 35 45 15 85 5 95 25]

Output :Sorted Array: [5 15 25 35 45 65 85 95]

Conclusion : Thus, we implemented Insertion sort and Shell sort to sort the given array in ascending order.

Experiment No. 6

Aim:

Implementation of quick sort in python

Objectives:

- a) To understand quick sort
- b) To understand divide and conquer

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.

Theory :

Quick Sort:

The basic version of quick sort algorithm was invented by C. A. R. Hoare in 1960 and formally introduced quick sort in 1962. It is based on the principle of divideand- conquer. It has two phases:

The quick sort partitions an array and then calls itself recursively twice to sort the resulting two subarray. This algorithm is quite efficient for large sized data sets as its average and worst case complexity are of $O(n\log n)$ where n are no. of items.

Most of the work is done in the partition phase - it works out where to divide the work. The sort phase simply sorts the two smaller problems that are generated in the partition phase.

:

This makes Quicksort a good example of the divide and conquers strategy for solving problems. This is similar in principle to the binary search. In the quicksort, we divide the array of items to be sorted into two partitions and then call the quicksort

procedure recursively to sort the two partitions, i.e. we divide the problem into two smaller ones and conquer by solving the smaller ones.

Algorithm:

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Input :Unsorted Array: [65 35 45 15 85 5 95 25]

Output :Sorted Array: [5 15 25 35 45 65 85 95]11 12 13 15

Conclusion : Thus, we implemented quick sort the given array in ascending order.

Experiment No. 7

Aim:

Implement Singly Linked List in C++

Objectives:

- a) To understand the concept of linked list data structure
- b) To understand
- c) To understand types of linked list
- d) To understand different operations on singly linked list(SLL).

Problem Statement:

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and 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:

1. Add and delete the members as well as president or even secretary.
2. Compute total number of members of club
3. Display members
4. Two linked lists exist for two divisions. Concatenate two lists.

Theory :

Linked List: A linked list is a linear data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a data and a reference (in other words, a link) to the next node in the sequence; more complex variants add additional links. This structure allows for efficient insertion or removal of elements from any position in the sequence. Each element (node) of a list comprises of two items reference/link to the next node. The last node has a reference to NULL. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a NULL reference.

Difference between array and linked list:

Array

- a) Arrays are linear data structures (for accessing and for storing in memory).
- b) Size of array is fixed.
- c) Static data structure.
- d) Array elements cannot be added, deleted once it is declared.
- e) Array elements can be modified easily by identifying the index value.
- f) Array are not difficulty to sort.
- g) Easy to locate the desired element.
- h) Memory allocation at the compilation time.
- i) Basically two types are there: dimension , two dimension.

Linked List

- a) Linked lists are linear for accessing, and non-linear for storing in memory.
- b) Size of linked list is not fixed.
- c) Dynamic data structure.
- d) The nodes in the linked list can be added and deleted from the list.
- e) It is a complex process for modifying the node in a linked list.
- f) Linked list are very difficult to sort.
- g) Can not immediately locate the desired element. Ne the whole list to reach that element.
- h) Memory allocation at the run time.
- i) Basically 3 types are there: doubly and circular.

Types of linked list;

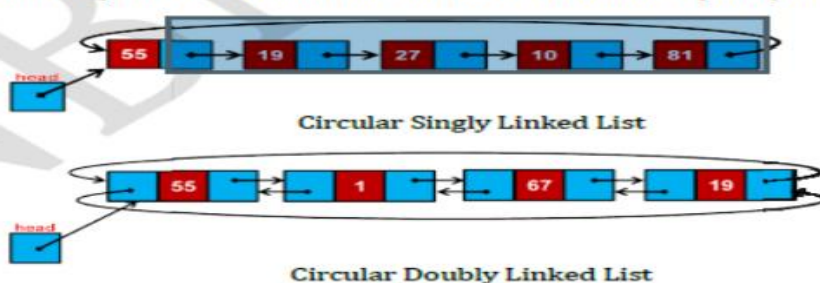
- a) **Singly Linked List:** Item Navigation is forward only.



- b) **Doubly Linked List:** Items can be navigated forward and backward way.



- c) **Circular Linked List:** Last item contains link of the first element as next (in SLL and DLL) and first element has link to last element as prev (in DLL).



Dynamic Memory:

In the programs, memory needs were determined before program execution by defining the variables needed. But there may be cases where the memory needs of a program can only be determined during runtime. For example, when the memory needed depends on user input. On these cases, programs need to dynamically allocate memory, for which the C++ language integrates the operators new and delete.

Operator new :

Dynamic memory is allocated using operator new. new is followed by a data type specifier and, if a sequence of more than one element is required, the number of these within

brackets []. It returns a pointer to the beginning of the new block of memory allocated. Its syntax is:

| | |
|---|---|
| Syntax: <code>pointer = new type;</code> or <code>pointer=new type[number_of_elements]</code> | Example: <code>int * ptr;</code> <code>ptr = new int [5];</code> |
|---|---|

In this case, the system dynamically allocates space for five elements of type int and returns a pointer to the first element of the sequence, which is assigned to ptr (an integer pointer). Therefore, ptr now points to a valid block of memory with space for five elements of type int.

Operator delete:

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator delete, whose syntax is:

| | |
|--|--|
| Syntax: <code>delete pointer;</code> or <code>delete[] pointer;</code> | Example: <code>delete ptr; //if ptr is pointer to variable</code> or <code>delete[] ptr; //if ptr is pointer to array</code> |
|--|--|

Algorithm:

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Time Complexity:

Insert / Delete at beginning: $O(1)$

Insert / Delete at end: $O(n)$

Search: $O(n)$

Indexing: $O(n)$

Input: Enter Post to be deleted

If 'p' then delete president from SLL (If president present).

If 's' then delete secretary from SLL (If secretary present).

If 'm' then ask name, then delete member with given name from SLL (If member with given name present).

Output:

Display student list (if deletion is successful)

or

President/Secretary/Member to be deleted not present in SLL

Conclusion : Thus, we implemented linear-dynamic data structure singly linked list to store students/members of student's club named 'Pinnacle Club'.

Experiment No. 8

Aim:

Perform different operation on Singly Linked List in C++

Objectives:

- a) To understand the concept of linked list data structure.
- b) To understand different operations on singly linked list (SLL).

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: Write C/C++ program to store two sets using linked list, compute and display:

- a. Set of students who like either vanilla or butterscotch or both
- b. Set of students who like both vanilla and butterscotch
- c. Number of students who like neither vanilla nor butterscotch

Theory:

Operations on set:

Definition: Let A and B be subsets of a set U.

Union of A and B: $A \cup B = \{x \in U \mid x \in A \text{ or } x \in B\}$

Union of the sets A and B, denoted $A \cup B$, is the set of all objects that are a member of A, or B, or both. The union of $\{1, 2, 3\}$ and $\{2, 3, 4\}$ is the set $\{1, 2, 3, 4\}$. We can represent the above set in terms of programming using array data structure as follows:

$A[] = \{1, 2, 3\}$ and $B[] = \{2, 3, 4\}$

Suppose C is set for storing union result.

So therefore union is $C[] = \{1, 2, 3, 4\}$.

Intersection of A and B: $A \cap B = \{x \in U \mid x \in A \text{ and } x \in B\}$

Intersection of the sets A and B, denoted $A \cap B$, is the set of all objects that are members of both A and B. The intersection of $\{1, 2, 3\}$ and $\{2, 3, 4\}$ is the set $\{2, 3\}$

. Intersection using array:

$A[] = \{1, 2, 3\}$ and $B[] = \{2, 3, 4\}$

Suppose C is set for intersection union result.

So therefore intersection is $C[] = \{2, 3\}$.

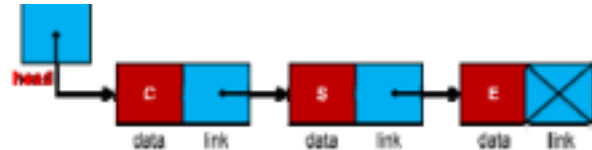
Difference of B minus A: $B - A = \{x \in U \mid x \in B \text{ and } x \notin A\}$

The difference of two sets B and A is the collection of objects in B that are not in A.
. The difference is written $B - A$.

Complement of A: $A^c = \{x \in U \mid x \notin A\}$

The complement of a set A is the collection of objects in the universal set that are not in A.

Linked List: A linked list is a linear data structure consisting of a group of nodes which together represent a sequence.



Following are the basic operations supported by a list.

- a) Insertion
- b) Deletion –
- c) Display –
- d) Search – search an element using given key.
- e) Delete – delete an element using given key.

Algorithm:

The pseudocode can be described as follows:

(Write your own algorithm/pseudocode)

Time Complexity:

Insert / Delete at beginning: $O(1)$

Insert / Delete at end: $O(n)$

Search: $O(n)$

Indexing: $O(n)$

Test Cases: Enter how many students: 4

Enter Roll: 25

Likes Vanilla (yes=1 / no=0):1

Likes Butterscotch (yes=1 / no=0):1

Enter Roll: 26

Likes Vanilla (yes=1 / no=0):1

Likes Butterscotch (yes=1 / no=0):0

Enter Roll: 27
Likes Vanilla (yes=1 / no=0):0
Likes Butterscotch (yes=1 / no=0):1
Enter Roll: 28
Likes Vanilla (yes=1 / no=0):0
Likes Butterscotch (yes=1 / no=0):0
Set of students who like either vanilla or butterscotch or
both = { 25, 26, 27 }
Set of students who like both vanilla and
butterscotch = { 25 }
Number of students who like neither vanilla nor
butterscotch = 1

Conclusion : Thus, we implemented linear-dynamic data structure singly linked list to store two sets and have performed various operations on it.

Experiment no.9

Aim: Implement Stack ADT in C++ using array

Objectives:

- To understand the concept of stack using array.
- To understand various stack operations.
- To understand use of stack to check string palindrome.
- To understand the use of stack to remove punctuation and to reverse the given string.

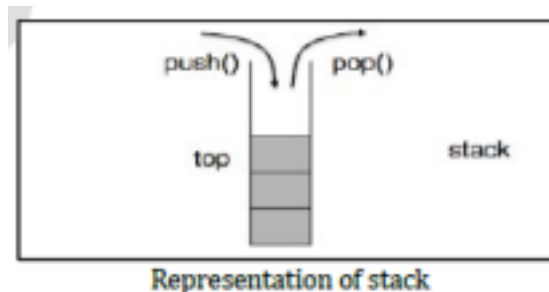
Problem Statement:

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-

- To print original string followed by reversed string using stack
- To check whether given string is palindrome or not

Theory :

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). In Stack all Operations such as Insertion and Deletion are permitted at only one end called Top.



Mainly the following basic operations are performed in the stack:

- **Push():** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop():** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.

- isFull() – check if stack is full.
- isEmpty() – check if stack is empty.

Implementation:

There are two ways to implement a stack:

- Using array
- Using linked list

Array Implementation of stack:

| | |
|--|--|
| <pre>void push(int x) { if(isFull()) { cout << "Stack Overflow"; } else { a[++top] = x; cout << "Element Inserted"; } }</pre> | <pre>int pop() { if(isEmpty()) { cout << "Stack Underflow"; return 0; } else { int d = a[top--]; return d; } }</pre> |
|--|--|

| | |
|--|---|
| <pre>bool isEmpty() { if(top < 0) return 1; else return -1; }</pre> | <pre>bool isFull() { if(top >= Max) { return 1; } else return -1; }</pre> |
|--|---|

Algorithm:

(Write your own algorithm/pseudocode)

Test Cases: Take following cases to check whether the string is palindrome or not: 1. Input: Enter a sting: madam

Output: Sting is a palindrome,
reverse of string is: madam

Input: Enter a string: Data Structures Laboratory

Convert this string to desired format (i.e. remove all spaces, punctuations and convert uppercases into lowercases) to check for palindrome as following,

datastructureslaboratory
now check for palindrome,

Output: Sting is NOT a palindrome,
reverse of string is: yrotarobalserutcurtsatad

Conclusion : Thus, we implemented various stack operation to check string palindrome, to reverse the given string.

Experiment No. 10

Aim: Implement C++ program for expression conversion as infix to postfix and its evaluation using stack.

Objective:

- a) To understand the use of stack.
- b) To understand expression conversion as infix to postfix using stack.
- c) To understand postfix expression evaluation using stack.

Problem Statement:

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

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

Theory:

Stack is mainly used for expression conversion and expression evaluation purpose.

In any programming language, if we want to perform any calculation or to frame a condition etc., we use a set of symbols to perform the task. These set of symbols makes an expression.

An expression can be defined as follows:

An **expression** is a collection of operators and operands that represents a specific value. In above definition, **operator** is a symbol which performs a particular task like arithmetic operation or logical operation or conditional operation etc.,

Operands are the values on which the operators can perform the task. Here operand can be a direct value or variable or address of memory location.

Based on the operator position, expressions are divided into three types. They are as follows:

- Infix Expression
- Postfix Expression
- Prefix Expression

Infix Expression

In infix expression, operator is used in between operands.

The general structure of an Infix expression is as follows...

Operand1 Operator Operand2

Example : a + b

Postfix Expression

In postfix expression, operator is used after operands. We can say that **"Operator followsthe Operands"**.

The general structure of Postfix expression is as follows...

Operand1 Operand2 Operator

Example : a b +

Prefix Expression

In prefix expression, operator is used before operands. We can say that **"Operandsfollows the Operator"**.

The general structure of Prefix expression is as

Operator Operand1 Operand2

Example : + a b

Any expression can be represented using the above three different types of expressions. And we can convert an expression from one form to another form like **Infix to Postfix, Infix to Prefix, Prefix to Postfix** and vice versa.

Infix to postfix conversion:

1. Scan through an expression, getting one token at a time.
2. Fix a priority level for each operator. For example, from high to low:
 3. - (unary negation)
 2. * /
 1. + - (subtraction)
3. Thus, high priority corresponds to high number in the table. If the token is an operand, do not stack it. Pass it to the output.
4. If token is an operator or parenthesis, do the following:
 - i. Pop the stack until you find a symbol of lower priority number than the current one. An incoming left parenthesis will be considered to have higher priority than any other symbol. A left parenthesis on the stack will not be removed unless an incoming right parenthesis is found.The popped stack elements will be written to output.

Convert the given infix expression I: $8-2+(3*4)/2^2$

Append right parenthesis) at end of I: $8-2+(3*4)/2^2)$

| Character from I | Stack | Postfix Expression P |
|------------------|---------|----------------------|
| | (| |
| 8 | (| 8 |
| - | (- | 8 |
| 2 | (- | 8 2 |
| + | (+ | 8 2 - |
| (| (+ (| 8 2 - |
| 3 | (+ (| 8 2 - 3 |
| * | (+ (* | 8 2 - 3 |
| 4 | (+ (* | 8 2 - 3 |

Fig: Infix to postfix conversion

- Stack the current symbol.
- If a right parenthesis is the current symbol, pop the stack down to (and including) the first left parenthesis. Write all the symbols except the left parenthesis to the output (i.e. write the operators to the output).
- After the last token is read, pop the remainder of the stack and write any symbol (except left parenthesis) to output.

Postfix Evaluation:

Postfix Expression: $82-34*22^{\wedge}/+$

Algorithm: Algorithm for Infix to Postfix conversion:

(Write your own algorithm/pseudocode)

Algorithm for evaluation of postfix expression

(Write your own algorithm/pseudocode)

Test Cases: For conversion of expression from infix to postfix:

Input: $(A + B)$

Output: $A B +$

For evaluation of postfix expression:

Input: $A B *$

Enter A: 4

Enter B: 5

Output: 20

Applications:

Conclusion:

Thus, we implemented expression conversion as infix to postfix and its evaluation using stack.

Outcome:

- Understanding the use of stack.
- Understanding of expression conversion as infix to postfix using stack.
- Understanding the postfix expression evaluation using stack.

Questions:

- 1) What is expression?
- 2) Explain different type of expression.
- 3) Explain Infix to postfix conversion.
- 4) Explain Postfix evaluation.

Experiment No. 11

Aim: C++ program for simulating job queue.

Objective:

- a) To understand the concept of queue.
- b) To understand various operation of queue.
- c) To understand use of various operations of queue using array.
- d) To understand the concept and use of priority queue.

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.

Theory:

Queue is a linear structure which follows a particular order in which the operations are performed. The order is **First In First Out (FIFO)**. A good example of queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Array implementation Of Queue

For implementing queue, we need to keep track of two indices, front and rear. We enqueue an item at the rear and dequeue an item from front.



Figure 1: Representation of queue

Operations on Queue:

Mainly the following four basic operations are performed on queue:

- **enqueue():** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
- **dequeue():** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
- **isfull():** checks if queue is full.
- **isempty():** checks if queue is empty

| | |
|---|--|
| isfull() Operation: bool isfull() { if(rear == MAXSIZE - 1) return true; else return false; } | isempty() Operation : bool isempty() { if(rear==front-1) return true; else return false; } |
|---|--|

| | |
|---|--|
| Enqueue Operation(i.e. insertion): int enqueue(int data) if(isfull()) return 0; rear = rear + 1; queue[rear] = data; return 1; | Dequeue Operation(i.e deletion): int dequeue() if(isempty()) return 0; int data = queue[front]; front = front + 1; return data; |
|---|--|

Priority Queue is more specialized data structure than Queue. Like ordinary queue, priority queue has same method but with a major difference. In Priority queue items are ordered by key value so that item with the lowest value of key is at front and item with the highest value of key is at rear or vice versa

A priority queue is used to handle the job queue of an operating system. So for the implementation of jobs of operating system priority queue is used. As the jobs are assigned in queue depending upon their priority

Priority Queue is Collection of entities or elements in which :

- **Addition** of **element** is done on **basis of priority**.
- **Removal** of **element** is done at **FRONT**.

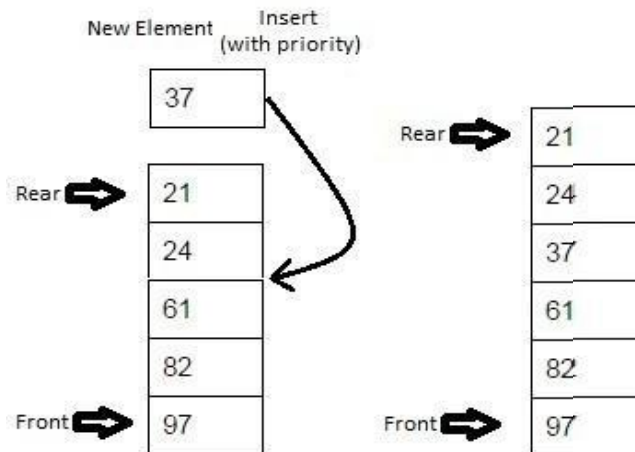


Figure 2: Insertion of element in priority queue depending upon priority.

Application of Queue-Queue is used when things don't have to be processed immediately, but have to be processed in **First InFirst Out** order like Breadth First Search. This property of Queue makes it also useful in following kind of scenarios.

- 1) When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.
- 2) When data is transferred asynchronously (data not necessarily received at same rate assent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

Algorithm: Enqueue Operation
(Write your own algorithm/pseudocode)

Dequeue Operation
(Write your own algorithm/pseudocode)

Insertion operation of priority queue:

(Write your own algorithm/pseudocode)

Test Cases: Enter how many jobs: 5

Enter Job (job priority 1-9) : 7

Enter Job (job priority 1-9) : 3

Enter Job (job priority 1-9) : 4

Enter Job (job priority 1-9) : 1

Enter Job (job priority 1-9) : 2

Dequeue jobs:

Job: 1

Job: 2

Job: 3

Job: 4

Job: 7

Applications:

Conclusion:

Thus, we implemented simulation of jobs using priority queue.

Outcome:

- Understanding the concept of queue.
- Understanding the various operation of queue.
- Understanding use of various operations of queue using array.
- Understanding the use of priority queue.

Questions:

1. Explain queue ADT.
2. Explain priority queue ADT.
3. Explain array representation of queue.
4. What are the applications of priority queue?

Experiment No. 12

Aim: Deque implementation in C++.

Objective:

- a) To understand the concept of deque.
- b) To understand various deque operations.
- c) To understand the use of various deque operation.

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.

Theory:

A **deque**, also known as a double-ended queue, is an ordered collection of items of similar types. It has two ends, a front and a rear. It allows inserting and deleting from both ends means items can be added or deleted from the front or rear end. Deque can behave like a queue by using only `enq_front` and `deq_front`, and behaves like a stack by using only `enq_rear` and `deq_rear`.

Double Ended Queue can be represented in two ways, those are as follows...

- Input Restricted Double Ended Queue
- Output Restricted Double Ended Queue

Input Restricted Double Ended Queue

In input restricted double ended queue, the insertion operation is performed at only one end and deletion operation is performed at both the ends.

Output Restricted Double Ended Queue

In output restricted double ended queue, the deletion operation is performed at only one end and insertion operation is performed at both the ends.

Various operations of deque:

- Insert element at rear
- Insert element at front

- Remove element at front
- Remove element at rear

Algorithm: Algorithm for Insertion of element from front:

(Write your own algorithm/pseudocode)

Algorithm for Insertion of element from rear:

(Write your own algorithm/pseudocode)

Algorithm for deletion of element from front:

(Write your own algorithm/pseudocode)

Algorithm for deletion of element from rear:

(Write your own algorithm/pseudocode)

Test Cases: Insert and delete the element in queue from rear end or front end.
Menu driven program with four choice:

1. Insertion from rear end
2. Insertion from front end
3. Deletion from front end
4. Deletion from rear end

Input : Element to be inserted or deleted

Output : Successful implementation of operation(i.e. insertion or deletion as per userchoice)

Applications:

Input:

Output:

Conclusion:

Thus, we implemented various deque operations using array.

Outcome:

Understanding the concept of deque.

- Understanding various deque operations.
- Understanding the use of various deque operation.

Questions:

- 1) Explain Deque ADT.
- 2) Explain concept of deque.
- 3) What are the applications of deque?
- 4) Differentiate queue, Deque and Priority queue

Experiment No. 13

Aim: C++ program for simulating circular queue.

Objective:

- To study Circular Queue as data structure.
- To understand implementation of Circular Queue and perform various operations on it.

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.

Theory:

Circular Queue

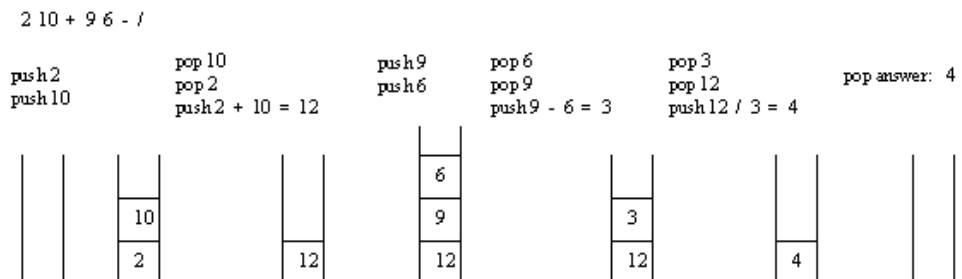
In a normal Queue Data Structure, we can insert elements until queue becomes full. But once if queue becomes full, we can't insert the next element until all the elements are deleted from the queue. For example consider the queue below:

This situation also says that Queue is Full and we can't insert the new element because, 'rear' is still at last position. In above situation, even though we have empty positions in the queue we can't make use of them to insert new element. This is the major problem in normal queue data structure. To overcome this problem we use circular queue data structure.

What is a Circular Queue?

A Circular Queue can be defined as follow:

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.



Algorithm: Implementation of Circular Queue:

(Write your own algorithm/pseudocode)

enQueue(value) – Inserting value into the Circular Queue:

(Write your own algorithm/pseudocode)

deQueue() – Deleting a value from the Circular Queue:

(Write your own algorithm/pseudocode)

display() – Displays the elements of a Circular Queue:

(Write your own algorithm/pseudocode)

Test Cases: -----PizzaParlor Menu -----

1. Place Order
2. Serve Order
3. Display Orders
4. Exit

Enter your choice: 1

Enter Order ID:1

Enter Orderer Name: ABX Pizza

Order Placed Successfully

Input:

Output:

Conclusion:

Thus, we implemented Circular Queue and perform various operations of it

Outcome:

- Understanding the concept of circular queue.
- Implement Circular Queue and perform various basic operations on it.

Questions:

1. What are the advantages of circular queue over linear queue.
2. Explain circular queue ADT.