# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



### **DATA STRUCTURES (23CS3PCDST)**

## Submitted by

SHANLKAR SHIVAPPA PUJAR (1BM23CS309)

in partial fulfillment for the award of the degree of BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING (Autonomous Institution under VTU) BENGALURU-560019 September 2024-January 2025

# B. M. S. College of Engineering, Bull Temple Road, Bangalore 560019 (Affiliated To Visvesvaraya Technological University, Belgaum) Department of Computer Science and Engineering



This is to certify that the Lab work entitled "DATA STRUCTURES" carried out by SHANKAR SHIVAPPA PUJAR(1BM23CS309), who is bonafide student of B. M. S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - (23CS3PCDST)work prescribed for the said degree.

Geetha N	Dr. Kavitha Sooda	
Assistant Professor	Professor and Head	
Department of CSE	Department of CSE	
BMSCE, Bengaluru	BMSCE, Bengaluru	

# **Index Sheet**

Sl.	Experiment Title	Page No.
No.		
1	Stack Implementation	4-9
2	Infix to Postfix Expression	10-15
3	Linear Queue	16-22
4	Circular Queue	23-30
5	Single Linked List (Insertion & Deletion)	31-53
6	Single Linked List with Operations	54-72
7	Single Linked List to Simulate Stack & Queue Operations	62-72
8	Doubly Linked List	73-83
9	Binary Search Tree	83-88
10	Traverse a Graph using BFS Method	89-96

**GITHUB LINK:** https://github.com/shankar045/DSA\_LAB.git

### Lab program 1:

Write a program to simulate the working of stack using an array with the following: a) Push b) Pop c) Display. The program should print appropriate messages for stack overflow, stack underflow

#### **PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
int stack[SIZE];
int top = -1;
void push(int element) {
  if (top == SIZE - 1) {
     printf("Stack Overflow! Cannot push %d\n", element);
  } else {
     top++;
     stack[top] = element;
     printf("%d pushed to stack\n", element);
  }
}
int pop() {
  if (top == -1) {
     printf("Stack Underflow! No element to pop\n");
     return -1;
  } else {
```

```
int poppedElement = stack[top];
     top--;
     return poppedElement;
  }
}
int peek() {
  if (top == -1) {
     printf("Stack is empty \n");\\
     return -1;
  } else {
     return stack[top];
  }
}
int isEmpty() {
  return top == -1;
}
int isFull() {
  return top == SIZE - 1;
}
void display() {
  if (top == -1) {
```

```
printf("Stack is empty\n");
  } else {
     printf("Stack elements are:\n");
     for (int i = top; i >= 0; i--) {
       printf("%d\n", stack[i]);
     }
  }
int main() {
  push(10);
  push(20);
  push(30);
  push(40);
  push(50);
  printf("\nTop element is: %d\n", peek());
  printf("Is stack full? %s\n", isFull() ? "true" : "false");
  printf("Is stack empty? %s\n", isEmpty() ? "true" : "false");
  printf("\nPopped element: %d\n", pop());
  printf("Popped element: %d\n", pop());
  display();
  return 0;
```

stackusingarra 10 pushed to stack 20 pushed to stack 30 pushed to stack 40 pushed to stack Stack Overflow! Cannot push 50 Top element is: 40 Is stack full? true Is stack empty? false Popped element: 40 Popped element: 30 Stack elements are: 20 10 Process returned 0 (0x0) execution time: 0.008 sPress ENTER to continue.

```
1) State operation 100) progress 10 implement. Steede lesing com
   # enclude Ksidio.hb
  of define MAN 5
  end Stanc (Mar);
  oul top= -1;
  coid push (one value) &
      if (top = : mmx-2) {
      printy (" steece overflow !" cantsput 'td | ", centis)
     Elser
       Steed [ top]: realue;
        prints ("y, d pushed to two steeck ", walne)
   uoid popu){
       )( [ = = ])(
        prints (" steele under flow! no element to popla")
                           (colles)
    Else 1
      proof (" 1. d popped from the steech In", steech [top]
  word display of
     7 (top == -0) /
```

```
print ("Stack is Empty (")))
                                                            yeturno.
  print I stare Elements are: ");
                                                            I defueld:
                                                                printy ("Encealed choice");
dor (not iso, icz top; it) (
     print (" Y.d", (tack(i));
   printy (" | " ).
  fut mars!) {
                                                              OUTPUT
    int charce walue;
       princy [" In , purh no , pop Ins , Display no : Exist | "")
    kehile (1) (
                                                             To perhed to steene
                                                             20 pulsed to Hear
       print (" Enter your choice ")
                                                             30 pulsed to stock
                                                             40 pushed to steads
        Say (" +d", schosce);
        Switch (chow) {
                                                              Top stace over flow
                                                               top elements 40
          CO41 1:
           Printy ("Exter realm to puth :");
                                                                Stark feull: false
           Slang (17.d", 4 walue):
                                                                popel Clement's 40.
            push (walu);
                                                                 Dopped clement & 30
           brease;
                                                                 Speec elements and
          casta:
                                                                  20
            pop();
                                                                  20
            break
           (asi)
              disply ()
               great,
```

### Lab program 2:

Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int precedence(char c) {
  if (c == '^')
     return 3;
  else if (c == '*' \parallel c == '/')
     return 2;
  else if (c == '+' \parallel c == '-')
     return 1;
   return -1;
char associativity(char c) {
  if (c == '^{\prime})
     return 'R';
  return 'L';
```

```
}
void infixToPostfix(char* exp) {
  int len = strlen(exp);
  char* result = (char*)malloc((len + 1) * sizeof(char));
  char* stack = (char*)malloc(len * sizeof(char));
  if (!result || !stack) {
     printf("Memory allocation failed\n");
     return;
  }
  int resultIndex = 0, stackIndex = -1;
  for (int i = 0; i < len; i++) {
     char c = \exp[i];
     if (isalnum(c)) {
       result[resultIndex++] = c;
     }
     else if (c == '('))
       stack[++stackIndex] = c;
     }
     else if (c == ')') {
       while (stackIndex \geq 0 \&\& stack[stackIndex] != '(') {
          result[resultIndex++] = stack[stackIndex--];
```

```
}
       stackIndex--; // Remove '(' from stack
     }
     else {
       while (stackIndex \geq 0 \&\&
            (precedence(c) < precedence(stack[stackIndex]) ||</pre>
            (precedence(c) == precedence(stack[stackIndex]) &&
             associativity(c) == 'L'))) {
          result[resultIndex++] = stack[stackIndex--];
        }
       stack[++stackIndex] = c;
  }
  while (stackIndex \geq 0) {
     result[resultIndex++] = stack[stackIndex--];
  }
  result[resultIndex] = '\0';
  printf("Postfix Expression: %s\n", result);
  free(result);
  free(stack);
int main() {
```

}

```
char exp[] = "a+b*(c^d-e)^(f+g*h)-i";
printf("Infix Expression: %s\n", exp);
infixToPostfix(exp);
return 0;
}
```

```
Infix Expression: a+b*(c^d-e)^(f+g*h)-i
Postfix Expression: abcd^e-fgh*+^*+i-
Process returned 0 (0x0) execution time : 0.006 s
Press ENTER to continue.
```

```
2> was to convol agreen realist presenthsized futinia aisthm.
 - tic Expusion to payin Empision The Expression consist
  I songle character operands and borrony operators + (plus)
   - (monus) & (multiply) and / (dietale)
  program
   # mchede < stole h 6
   At melude (stallib. h)
  Amelacke (conte h)
  int pec (change) {
     1 (c: : , V.)
         1ctam 3
     502 1 (C==1, 11 C== , K)
         refamo.
     Elsey (C: 5.4. |10 = 5.-)
         return ?!
    Else refor -27,}
   chang associativity (chance) {
    H(c=: 'N')
      yetunk"
    refun'L".
coesd rulinia Topost for (court Charts) (
     and len- stolenes);
```

```
char result = (char x) malloc (lents):
Chagk stack = (Chank ) multor (Len);
 not result Inde = 0;
 ant stack Index 5-2,
  is ( | result ! ! Stack ) {
    Drink (" Memory allocation failed [ ""];
 for (nuti =0; icken; i+1)
 Charge - Stil;
   il ((c)= 0,41 c7= 5,11 (c)= 1,44 c7= 5,11 c>=0,44 60
          regul (resultedeatt ]= C
  Elseil (2: :1())
    Ste Stelle [H Stell Endra]=c;
  Strail (6 = ( ) {
  Else 1/(c==')') 6
      While (Steeck Index >= 0 ff Steece [Steece Index ] = (1) f
       I guell [ Rguel Just 34 + ]: Stone [ Stone [ Stone Indo - ]
       Steek Ended . --;
    such
      natife (Steecle Index> = 0 11 (pue Co) 2 prac (Steere (5to
       Ender J) 11 (Duc (C) = : prec (Steer (Steer Ender)
```

```
As associatively (c)=: 2/)1){
     result (reput Endex ++ ] = steel [steele gratex -- ];
   Steeck ( + + Steeck Irds) = c;
Klhile (Stack Enders >= 0) d
the Steek
    result (result Index ++ ]: Steech: TS teach Endere - ];
result result Inda J: 10;
 Printy ("%.s/n", requet );
free (resut);
 free (Steet); }
 mt man () {
   charexpl): "a + 6 (c/d -e) " (f+g*h)-: ".
  enfinia To portito (Exp);
   returno
```

# Lab program 3:

WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display

The program should print appropriate messages for queue empty and queue overflow conditions

### **PROGRAM:**

#include <stdio.h>

```
#include <stdlib.h>
#define QUEUE_SIZE 10
int queue[QUEUE_SIZE];
int front = -1, rear = -1;
void insert(int item) {
  if (rear == QUEUE SIZE - 1) {
    printf("Queue Overflow! Cannot insert %d.\n", item);
    return;
  }
  if (front == -1)
```

```
front = 0;
  }
  rear++;
  queue[rear] = item;
  printf("Inserted: %d\n", item);
}
int delete() {
  if (front == -1 \parallel front > rear) {
     printf("Queue Underflow! Queue is empty.\n");
     return -1;
  }
  int deletedItem = queue[front];
  printf("Deleted: %d\n", deletedItem);
  front++;
  if (front > rear) {
     front = rear = -1;
  }
  return deletedItem;
```

```
void display() {
  if (front == -1 \parallel front > rear) {
     printf("Queue is empty.\n");
     return;
   }
  printf("Queue contains: ");
  for (int i = front; i \le rear; i++) {
     printf("%d ", queue[i]);
  }
  printf("\n");
}
int main() {
  int choice, item;
  while (1) {
     printf("\nQueue Operations:\n");
     printf("1. Insert\n");
     printf("2. Delete\n");
     printf("3. Display\n");
     printf("4. Exit\n");
     printf("Enter your choice: ");
```

```
scanf("%d", &choice);
     switch (choice) {
       case 1:
          printf("Enter the element to insert: ");
          scanf("%d", &item);
          insert(item);
          break;
       case 2:
          delete();
          break;
       case 3:
          display();
          break;
       case 4:
          printf("Exiting program.\n");
          exit(0);
          break;
       default:
          printf("Invalid choice! Please try again.\n");
     }
  }
return 0;
```

```
Queue Operations:
1. Insert
2. Delete
Display
4. Exit
Enter your choice: 1
Enter the element to insert: 10
Inserted: 10
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted: 10
Queue Operations:
1. Insert
2. Delete
Display
4. Exit
Enter your choice: 3
Queue is empty.
Queue Operations:
1. Insert
2. Delete
Display
4. Exit
Enter your choice: 4
Exiting program.
```

```
3) MIPP to stimulate the worksong gor quare g'intiger
  usme an array procede the fallowing operators musest
  deleat display.
  Franchesle (8-1410 b)
  # define over Size 10
   int item pout = 10, teat = - 1, 9[10]
   void mosest () d
   il (reat = que 8 me -1) {
    printy ("stack overflow! | ");
     return:
   Year + = 2)
   9 [rear] = item;
  int deleate () {
  il (bont) rear) {
  printy (" o vecis empty | 10-);
   return - 2;
return of [non +2];
boid display () {
int i
if (front > rear) (
printy (" Queue, is empty | nu?
```

```
return;
    points (" queue contains: | ");
    for ( := fout: iz = sear; :++) {
          print ("1.1 | ", 9 8 3)
   output:
   Options:
 1) Ensert
 2) Defeate
 3) print green
 4) Exist
 Select an option: 1
 enter the value to ment:10
 options:
 1) Insert
2) Defeate
3) print queue
4) Ex 251
select an option: 3
Current queue confung: 10
  options:
  1) Enseal
   e) pelete
   3) prut queue
        ENIST
```

# Lab program 4:

Write a program to simulate the working of a queue of integers using an

array. Provide the following operations

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue

overflow conditions

### **PROGRAMM:**

```
#include <stdio.h>
#define SIZE 5

int queue[SIZE];
int front = -1, rear = -1;

int is_full() {
    return ((rear + 1) % SIZE == front);
}

int is_empty() {
```

```
return (front == -1);
}
void insert(int value) {
  if (is_full()) {
     printf("Queue Overflow\n");
     return;
  if (front == -1) {
     front = rear = 0;
  } else {
     rear = (rear + 1) \% SIZE;
  }
  queue[rear] = value;
  printf("Inserted: %d\n", value);
}
void delete() {
  if (is_empty()) {
     printf("Queue Underflow\n");
     return;
  }
```

```
printf("Deleted: %d\n", queue[front]);
  if (front == rear) {
     front = rear = -1;
  } else {
     front = (front + 1) \% SIZE;
void display() {
  if (is_empty()) {
     printf("Queue is Empty\n");
     return;
  }
  printf("Queue: ");
  int i = front;
  while (1) {
     printf("%d ", queue[i]);
    if (i == rear) break;
     i = (i + 1) \% SIZE;
  }
  printf("\n");
```

```
int main() {
  int choice, value;
  while (1) {
     printf("\nCircular Queue Operations:\n");
     printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
     printf("Enter your choice: ");
     scanf("%d", &choice);
     switch (choice) {
       case 1:
          printf("Enter the value to insert: ");
          scanf("%d", &value);
          insert(value);
          break;
       case 2:
          delete();
          break;
       case 3:
          display();
          break;
       case 4:
          printf("Exiting...\n");
          return 0;
```

```
default:
    printf("Invalid choice, please try again.\n");
}
return 0;
}
```

# **OUTPUT 4:**

```
Circular Queue Operations:

    Insert

2. Delete
Display
4. Exit
Enter your choice: 1
Enter the value to insert: 5
Inserted: 5
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted: 5
Circular Queue Operations:

    Insert
    Delete

Display
4. Exit
Enter your choice: 3
Queue is Empty
Circular Queue Operations:

    Insert

2. Delete
Display
4. Exit
Enter your choice: 4
Exiting...
Process returned 0 (0x0)
                              execution time : 49.717 s
Press ENTER to continue.
```

21/10/24 to Stimulate Working & Circular - puece & intigers using an anny proceede the fallowing operation The progress Should print appropriate Messages for a greece empty and occur overflow condition Insert delete & Display progremo ! Honclude (Stdio. h) # define 8.20 10 out point reas = -1) } pas by selectie not is - full () { if (front == 0 88 decis sine -2)11 (sear = : (fout-s)) ( refam? sud: ( front = = -2) { return 1; } reof Insert ( But walke) 1 14 (is fuel) L Prints (" oceae overflow hi"); refurn'];

```
if (port = = -2) {
                                                     void display () L
   front = sears 0;
                                                      if (is empty ()) }
   else if (rears size-144 pout 12=0) (
                                                         printy (" Queue is Empty "); }
    rear o;
                                                         prints (" queue: ");
                                                           for (and 1: front; il: rear; i= (i+1)9. Suc)
  Else
                                                                  Bridg (" to d", que (13);
      rears (reasts)/ Size;
  Ource Event) = value;
   loosed deleter ) {
      if ( es Empty ()) {
     printel " Oucue is overflow | " ); }
                                                       output !
    prints (" Defete 4.4 M", que (front));
                                                      Circular greene operations:
   if (front = = rear) (
                                                       3) display
      front = rear = = = = = = =
                                                       4) Exist
      Else if fruit: (ou-1) L
                                                       Geter your choice: 1
                                                        Enter a realue tomsest: 5
        front = 04
                                                          Injusted 5
    front = (front + 1).1. Size;
                                                      Countary Outcome operations
                                                       12 Intent dele
```

Created green operations: York display () ( 1. Socar 2. Delete : + (11\_cmpby ()) ( 3. Display . I have been a prints (" arene & cospty"); } 4- 50:11 . Enter your choice :4 prints ( " glecce "); for (mit : - fout ; ; to mor; i= (i1)) /. Sine) ( - printo["1.d", Apreced[1]); concerta que operations: 1. Insect 2. pefete 3. Display 4, Exist enter your chair: 3 June Element: 5 Circles queue operations: 11 Enjor a. Delete 3- Diploy 4, 6211 Deleted 5

# Lab program 5 A:

Write a program to implement Singly Linked List with following

operations

- a) Create a linked list.
- b) Insertion of a node at first position, at any position and at end of list.
- c) Display the contents of the linked list

#### PROGRAMM:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node* next;
};

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
if (newNode == NULL) {
    printf("Memory allocation failed.\n");
    exit(1);
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}
void insertBeg(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  newNode->next = *head;
  *head = newNode;
}
void insertEnd(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head == NULL) {
    *head = newNode;
    return;
  struct Node* temp = *head;
```

```
while (temp->next != NULL) {
    temp = temp->next;
  }
  temp->next = newNode;
}
void displayList(struct Node* head) {
  if (head == NULL) {
    printf("The list is empty.\n");
    return;
  }
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d -> ", temp->data);
    temp = temp->next;
  }
  printf("NULL\n");
}
int main() {
  struct Node* head = NULL;
```

```
int choice, value;
while (1) {
  printf("\n1. Insert at beginning\n");
  printf("2. Insert at end\n");
  printf("3. Display the linked list\n");
  printf("4. Exit\n");
  printf("Enter your choice: ");
  scanf("%d", &choice);
  switch (choice) {
     case 1:
       printf("Enter value to insert at beginning: ");
       scanf("%d", &value);
       insertBeg(&head, value);
       break;
     case 2:
       printf("Enter value to insert at end: ");
       scanf("%d", &value);
       insertEnd(&head, value);
       break;
     case 3:
       printf("Linked List: ");
```

```
displayList(head);
break;
case 4:
    printf("Exiting...\n");
    exit(0);
default:
    printf("Invalid choice, please try again.\n");
}
return 0;
}
```

```
2. Insert at end
3. Display the linked list
4. Exit
Enter your choice: 1
Enter value to insert at beginning: 10

1. Insert at beginning
2. Insert at end
3. Display the linked list
4. Exit
Enter your choice: 2
Enter value to insert at end: 20

1. Insert at beginning
2. Insert at beginning
3. Display the linked list
4. Exit
Enter your choice: 3
Linked List: 10 -> 20 -> NULL

1. Insert at beginning
2. Insert at end
3. Display the linked list
4. Exit
Enter your choice: 3
Linked List: 10 -> 20 -> NULL

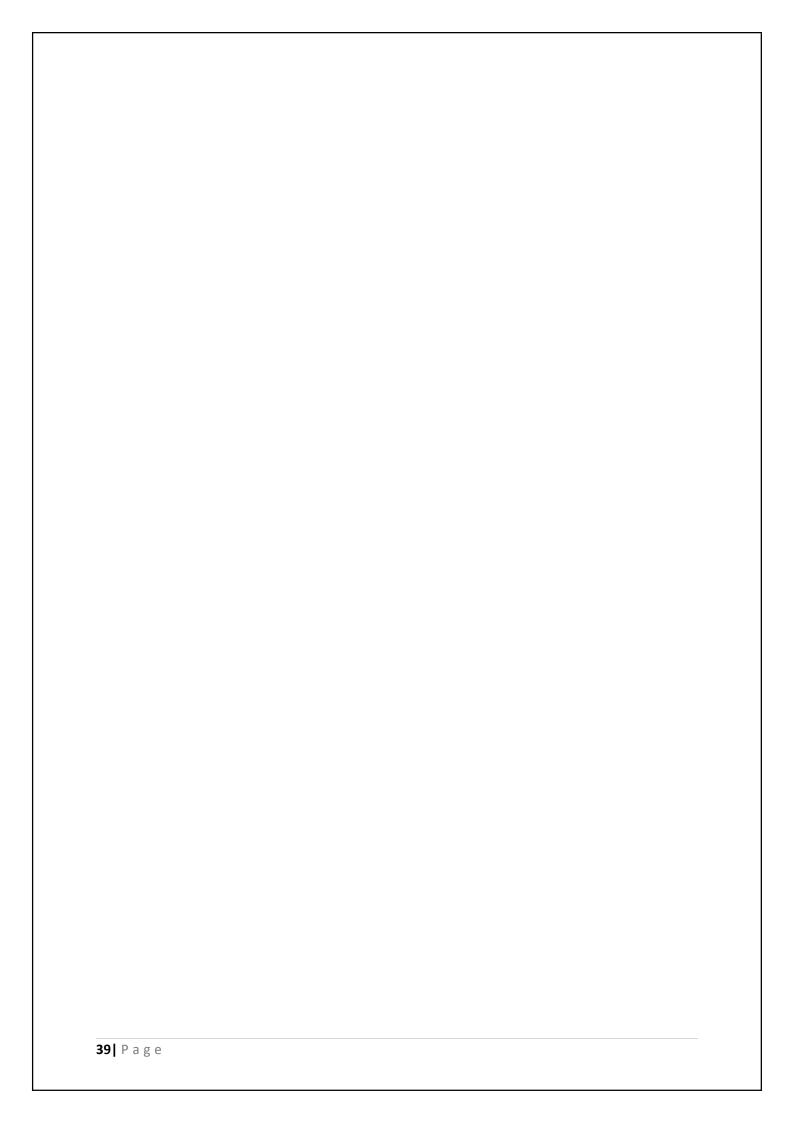
1. Insert at beginning
2. Insert at end
3. Display the linked list
4. Exit
Enter your choice: 4
Exiting...

Process returned 0 (0x0) execution time: 57.032 s
Press any key to continue.
```

```
00 Meet 40 14
ON WAR to Implement Souly linked list Klith fallowing
   Operation
    9) (reate a laked list
    5) Insertion guide citist position at end q list
     Diplay the content & the looked list
   program .
  Fractude (stdiah)
  Hanelusle (stallb.h)
  Struck Nade 1
      out data:
     Anut water next; }!
  Street Noder Create Node (rut data)
     Street Node new Abde = (Sheet Nocle * ) Malloc ( sore & (street rede))
       Newwoode -> data = data;
       new Mode - data = Nucl
       return New node;
 world onsort At Jugintning (Annel Moders head, ant data) (
     Street woole " New Mode =
          Create Node (data);
         neve mode > next = " head ;
         I head = Hew plade;
word anyouth End (Stant Noders land midate) {
         Stud mode , new mode =
           Create Mode (data);
```

```
if ( * head = = kever) {
 & head : New Mode;
  seturn)
Street Noder temp = x head;
   wells (temp > Next ) = Nucl) {
     temp = temp = neat;}
   temp - next = hew node;
  voed display (street Noch + head) {
        Street Nocle + term P, - houd;
      While (tempe New) {
       prints (" "rol", temp-> data); I
         temp stemp - next;
        prints (11 Muers)!
    output "
    1. Insert of segginning
    2. Engest at end
    3, Display
    4. exist
    Enter your Choice: 1
```

Menu 1 Inset at segenwong d. Insert out end 3. Display 11st d. Ex. st Enter your choice : 2 Enter data total: 20 fi hours quint Mem: is horn zarad + to are a good - Insert of segining ( tank & down to the segin by 1. Insert at end Precel made + trans Can. s. Diploy list Easy+ Enferyous choice: 20' i of ob and make it is in the Enter data 12 most end Enter data to onsest end: 40 : (" 18 19 ") storing Mena: Enfo your choice: 3 loubal 2014:20-5 10-540-3 Much



# Lab program 5 B:

Write a program to Implement Singly Linked List with following operations

- a) Create a linked list.
- b) Deletion of first element, specified element and last element in the list.
- c) Display the contents of the linked list.

#### **PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node* next;
};

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
```

```
printf("Memory allocation failed.\n");
    exit(1);
  }
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}
void insertBeg(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  newNode->next = *head;
  *head = newNode;
}
void insertEnd(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head == NULL) {
    *head = newNode;
    return;
  }
  struct Node* temp = *head;
  while (temp->next != NULL) {
```

```
temp = temp->next;
  }
  temp->next = newNode;
}
void deleteFirst(struct Node** head) {
  if (*head == NULL) {
    printf("The list is empty, nothing to delete.\n");
    return;
  }
  struct Node* temp = *head;
  *head = (*head)->next;
  printf("Deleted: %d\n", temp->data);
  free(temp);
}
void deleteLast(struct Node** head) {
  if (*head == NULL) {
    printf("The list is empty, nothing to delete.\n");
    return;
  }
  if ((*head)->next == NULL) {
```

```
printf("Deleted: %d\n", (*head)->data);
    free(*head);
    *head = NULL;
    return;
  struct Node* temp = *head;
  while (temp->next->next != NULL) {
    temp = temp->next;
  printf("Deleted: %d\n", temp->next->data);
  free(temp->next);
  temp->next = NULL;
}
void deleteSpecific(struct Node** head, int key) {
  if (*head == NULL) {
    printf("The list is empty, nothing to delete.\n");
    return;
  if ((*head)->data == key) {
    struct Node* temp = *head;
    *head = (*head)->next;
```

```
printf("Deleted: %d\n", temp->data);
    free(temp);
    return;
  struct Node* temp = *head;
  while (temp->next != NULL && temp->next->data != key) {
    temp = temp->next;
  }
  if (temp->next == NULL) {
    printf("Element %d not found in the list.\n", key);
    return;
  struct Node* toDelete = temp->next;
  temp->next = temp->next->next;
  printf("Deleted: %d\n", toDelete->data);
  free(toDelete);
}
void displayList(struct Node* head) {
  if (head == NULL) {
    printf("The list is empty.\n");
    return;
```

```
}
  struct Node* temp = head;
  while (temp != NULL) {
     printf("%d -> ", temp->data);
     temp = temp->next;
  }
  printf("NULL\n");
}
int main() {
  struct Node* head = NULL;
  int choice, value;
  while (1) {
     printf("\n1. Insert at beginning\n");
     printf("2. Insert at end\n");
     printf("3. Delete first element\n");
     printf("4. Delete last element\n");
     printf("5. Delete specific element\n");
     printf("6. Display the linked list\n");
     printf("7. Exit\n");
     printf("Enter your choice: ");
```

```
scanf("%d", &choice);
switch (choice) {
  case 1:
    printf("Enter value to insert at beginning: ");
    scanf("%d", &value);
    insertBeg(&head, value);
    break;
  case 2:
    printf("Enter value to insert at end: ");
    scanf("%d", &value);
    insertEnd(&head, value);
    break;
  case 3:
    deleteFirst(&head);
    break;
  case 4:
    deleteLast(&head);
    break;
  case 5:
    printf("Enter value to delete: ");
    scanf("%d", &value);
```

```
deleteSpecific(&head, value);
          break;
       case 6:
          printf("Linked List: ");
          displayList(head);
          break;
       case 7:
          printf("Exiting...\n");
          exit(0);
       default:
          printf("Invalid choice, please try again.\n");
     }
  }
  return 0;
}
```

# **OUTPUT:**

```
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 1
Enter value to insert at beginning: 10
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element6. Display the linked list
7. Exit
Enter your choice: 1
Enter value to insert at beginning: 20
1. Insert at beginning

    Insert at end
    Delete first element

4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 2
Enter value to insert at end: 30
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 6
Linked List: 20 -> 10 -> 30 -> NULL
```

```
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 3
Deleted: 20

    Insert at beginning
    Insert at end
    Delete first element
    Delete last element

5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 6
Linked List: 10 -> 30 -> NULL
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 4
Deleted: 30
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
```

Enter your choice: 6
Linked List: 10 -> NULL

```
1. Insert at beginning
2. Insert at end
3. Delete first element
4. Delete last element
5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 5
Enter value to delete: 10
Deleted: 10
1. Insert at beginning
2. Insert at end

    Delete first element
    Delete last element

5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 6
Linked List: The list is empty.

    Insert at beginning
    Insert at end
    Delete first element
    Delete last element

5. Delete specific element
6. Display the linked list
7. Exit
Enter your choice: 7
Exiting...
Process returned 0 (0x0)
                               execution time : 153.277 s
Press any key to continue.
```

plainte a progress to empliment Souply linked list help to factowing operations a) Create a linked list 6) Defetion of find element specified and last element c) Display the contents of the linked list # includedstaliohs # enclude & conil.h street model out data; Street rode \* neat Struct Node \* Greate Mode (out data) / Street Noder New Mode = (Struct Node\*) Marioc (Some of Structivale) new Node -> data = data; New Mode -> next = MULL' · yetum newpode 102d getd ( of next wode ex head . \* ont data) & Street moder new Mode = Create Mode (data); of ( & hiad set = HULC) ( \* fread sel stem Mode. Street Mode + last = \* head ref: kellile clast & next ! : wiech east : last - ment;

last of next = were readle; of temp== Kind ( prints ( " And teed vid to too elist of ", data); prints (" element you not find fur); word defetepart ( Street modern headray) ( return ! if ( head sy: Nuce) { provision rext tempes rout of the first the prints ("list is empty Norwing tobletete (mu) print (" Defetered element ( rd/u", Kg)) referm; Ince (temp); Street peak + temp + head soft; roid deletelest (start floder & head ref) & \* bead seg: ( \* bead reg) -> mest; 18 ( head of = Mille) & - 1 along states prints (" Defeted first element and In", temporate prints (" leiter empty / ") fre (femp). Void defete element Copyet moder - head my int Street Noder temp: A head my; of (femp) next = need 1111. Strue Noder temp = " head is of " print NULL printy ("Deleted - last element y.d/in", temp-solat b ( temp): Nove for temps data: = keysh Drinks (" Deteted specified eferent /dln", key)! gree (femp); A bread - ref mull; reform; defon; while (fempl = Mull off temp -> data] = Kook Struct peaces prev: pices; pulole (temper next) = Muccold pray: temp; temps temp -> Notty prive temp; eup : jeup - yeat

prints ["Defeted last element: "Id his, semps dated. prov-snest= Nucles free (temp); vord desplay lest (stouch mode \* node) of (node = + Nucle) L prints [" left is Empty 9 ""); refurn; prints (" linked lest"); Keleste (node] = Nucley & print ("/d -> ", node-sdata); node : node - s next; bright (MATTIN, ); output: , Enseut al beggmong 3, perete forst element 4. peteless by element 5, Defek sperger element. 6. Display tou lint solet 7. oxist our choice : à bognes 1921 - 12km - 2 - 91 - 27 - 3 - 20 - 20

I triest at beginning d. Insect cut and 3. Delege first element U. Defete last chand 5 pisplay son england ust 6. Display the finted list. 2. exist Ends your choice: 2 tupe , value to most at end, 30 7. Expest at beginning 2 Enset at end 4. Defete fost element. 5. Delete spanje element. 6, Display the linked list. De Graist Cuter your choir 6 light : 20-10-30-101e 1. Prosent at begginning 2, bysel atend 3, Defete, first clamp of beliefe last element S, Délete spenfer-elment 6, proper the encedard 7. Guilt Erko your cuoice:6 linced list: 10-30-pull

1. I need at largence 9 I . Injert of end & believe changed. 4. Delete Joy + clamed \$, Delete Specific cleans 7 seist Enteryour Choice 4 Deleted : 30 Enfect your Charce: 6 linked list to - pull Enter gour Choice is Enter water to defeted. 10 peleted: 10 Enter your choice : 6 looked Wil: The lost or conft. Enfet your choice ? Existing . . .

# LAB PROGRAM 6 A:

WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* next:
};
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if (newNode == NULL) {
    printf("Memory allocation failed!\n");
    return NULL;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
```

```
void append(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head == NULL) {
    *head = newNode;
    return;
  struct Node* temp = *head;
  while (temp->next != NULL) {
    temp = temp->next;
  temp->next = newNode;
}
void printList(struct Node* head) {
  if (head == NULL) {
    printf("List is empty\n");
    return;
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d -> ", temp->data);
    temp = temp->next;
  printf("NULL \n");
```

```
}
void sortList(struct Node* head) {
  if (head == NULL) {
     return;
  struct Node* i;
  struct Node* j;
  for (i = head; i\rightarrow next != NULL; i = i\rightarrow next) {
     for (j = i-next; j != NULL; j = j-next) {
       if (i->data > j->data) {
          int temp = i->data;
          i->data = j->data;
         j->data = temp;
void reverseList(struct Node** head) {
  struct Node* prev = NULL;
  struct Node* current = *head;
  struct Node* next = NULL;
  while (current != NULL) {
```

```
next = current->next;
    current->next = prev;
    prev = current;
    current = next;
  *head = prev;
void concatenateList(struct Node** head1, struct Node* head2) {
  if (*head1 == NULL) {
    *head1 = head2;
    return;
  }
  struct Node* temp = *head1;
  while (temp->next != NULL) {
    temp = temp->next;
  temp->next = head2;
}
int main() {
  struct Node* head1 = NULL;
  struct Node* head2 = NULL;
```

```
append(&head1, 10);
append(&head1, 20);
append(&head1, 30);
append(&head2, 5);
append(&head2, 15);
append(&head2, 25);
printf("List 1: ");
printList(head1);
printf("List 2: ");
printList(head2);
sortList(head1);
sortList(head2);
printf("Sorted List 1: ");
printList(head1);
printf("Sorted List 2: ");
printList(head2);
concatenateList(&head1, head2);
printf("Concatenated List: ");
```

```
printList(head1);

reverseList(&head1);

printf("Reversed Concatenated List: ");
printList(head1);

return 0;
}
```

# **OUTPUT:**

```
List 1: 10->20->30->null

List 2: 5->15->25->null

Sorted List 1: 10->20->30->null

Sorted List 2: 5->15->25->null

Concatenated List: 10->20->30->5->15->25->null

Reversed Concatenated List: 25->15->5->30->20->10->null

Process returned 0 (0x0) execution time: 0.000 s

Press any key to continue.
```

69) KIAS to implement Single looked list neath faccowing operations; Sost the looked list Reverse the Inted list Concatination & two disked lest Honclade Lettions # on clude Lotatio. ht typeday sine of Mode ( out data". Street Hoder next! ) (otable) ( 36) } Mode, Mode « create mode (out dater) f Mode , very voge = · ( Mode\* ) marloc (Sme of (Mode)); new Mode-s data = data: newprode-s rest = MULL! retura seme mode; dur : vig tolor 1551111 - 1214 (cold meet ( Node \* head, not data) f Moder new Mode = (seafe Mode (data)) if ( head = = NUCE) { there to have I had : new rode; Else ( Node \* temp = \* head! ketile (temp -> next ; = Nececi) { printy (" y.d " , head => date); read = head -snext!

```
prints (" MOLC |n")
word fort ( Mode + T head ) {
   Mode " ! * ; ;
    out comp:
 for (is * head; il = Nuce; is in mit) {
     for (3=1= mext; j |= Mucc; j=j=next)
     ig (in data) j->data) (
        temp : 1 - Jata;
        P - data = j - s data;

j - data = temp;
   world revenue (Moders head) (The makestation
       hode priv = kute " lanent = head,
     * wet : Muce;
        table ( ( carrent ! = NUCC) {
        nent = Current -+ rest;
        ( when - hest = Doer; many whole
        mer = current; (100m / horse)
         (uned : not : ton
        ) * head = pien;
                 the same of select
          Garage of the englast of spel
```

```
10 Ed Concatinate (moders board?),
  Nocle ** heed 2) 4
    ig ( heads = skell) {
      * head = x head d;
      I else {
      Moder temp = " heard ?"
       helie le temp + rext! = NEEC)
        terns - temp -> paxel;
         temp-s next = x head 2:
                 (otalina) story story + story
  output!
   1:(+): 10-) (0-) 30-) MIL . Mallana and
    list 2:5-3 15-5 28-1 MULL
     Storting fift?
     10-1 20-12-1 Nelle
      Revensing 1917
        30 -> 20 -> 10-> NULL
      Containing list & and list 2.
        30-120-10-15-7 15-3 25-1 Nauce
```

# LAB PROGRAM 6 B:

WAP to Implement Single Link List to simulate Stack & Queue Operations.

#### **PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* next;
};
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if (newNode == NULL) return NULL;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}
void push(struct Node** top, int data) {
  struct Node* newNode = createNode(data);
  if (!newNode) return;
  newNode->next = *top;
```

```
*top = newNode;
}
int pop(struct Node** top) {
  if (*top == NULL) return -1;
  struct Node* temp = *top;
  int poppedData = temp->data;
  *top = (*top)->next;
  free(temp);
  return poppedData;
}
int peek(struct Node* top) {
  if (top == NULL) return -1;
  return top->data;
}
int isStackEmpty(struct Node* top) {
  return top == NULL;
}
void enqueue(struct Node** front, struct Node** rear, int data) {
  struct Node* newNode = createNode(data);
  if (!newNode) return;
```

```
if (*rear == NULL) {
    *front = *rear = newNode;
    return;
  }
  (*rear)->next = newNode;
  *rear = newNode;
}
int dequeue(struct Node** front, struct Node** rear) {
  if (*front == NULL) return -1;
  struct Node* temp = *front;
  int dequeuedData = temp->data;
  *front = (*front)->next;
  if (*front == NULL) *rear = NULL;
  free(temp);
  return dequeuedData;
}
int peekQueue(struct Node* front) {
  if (front == NULL) return -1;
  return front->data;
}
int isQueueEmpty(struct Node* front) {
```

```
return front == NULL;
}
void printList(struct Node* head) {
  if (head == NULL) {
    printf("List is empty\n");
    return;
  }
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d -> ", temp->data);
    temp = temp->next;
  }
  printf("NULL\n");
}
int main() {
  struct Node* stackTop = NULL;
  struct Node* queueFront = NULL;
  struct Node* queueRear = NULL;
  push(&stackTop, 10);
  push(&stackTop, 20);
  push(&stackTop, 30);
```

```
printf("Popped from stack: %d\n", pop(&stackTop));
printf("Top element of stack: %d\n", peek(stackTop));
enqueue(&queueFront, &queueRear, 100);
enqueue(&queueFront, &queueRear, 200);
enqueue(&queueFront, &queueRear, 300);
printf("Dequeued from queue: %d\n", dequeue(&queueFront, &queueRear));
printf("Front element of queue: %d\n", peekQueue(queueFront));
printf("Stack elements: ");
printList(stackTop);
printf("Queue elements: ");
printList(queueFront);
return 0;
```

}

#### **OUTPUT:**

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value to push onto stack: 10
10 pushed onto the stack.

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value to push onto stack: 20
20 pushed onto the stack.

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value to push onto stack: 20
20 pushed onto the stack.

Menu:
1. Push (Stack)
3. Display Stack
4. Enqueue (Queue)
6. Display Queue
7. Exit
Enter your choice: 3
Stack: 20 -> 10 -> NULL

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Queue
7. Exit
Enter your choice: 3
Stack: 20 -> 10 -> NULL

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
```

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 2
20 popped from the stack.
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 3
Stack: 10 -> NULL
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 4
Enter value to enqueue into queue: 15
15 enqueued into the queue.
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
```

```
Enter your choice: 4
Enter value to enqueue into queue: 15
15 enqueued into the queue.
Menu:
1. Push (Stack)
2. Pop (Stack)

    Display Stack
    Enqueue (Queue)

5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 4
Enter value to enqueue into queue: 25
25 enqueued into the queue.
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 6
Queue: 15 -> 25 -> NULL
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 5
15 dequeued from the queue.
```

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 6
Queue: 25 -> NULL

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 7
Exiting...

Process returned 0 (0x0) execution time: 167.726 s
Press any key to continue.
```

```
b) WAP to implement Emple broked list to Stimulate
                                                   mode & femp = * femp;
Steech & quae operation:
                                                   out duto stemp - Lator
                                                   $ top = (xtop) -> rent.
                                                     force (femp)
Honclude & State hs
                                                     return datas
# onclude LStallib. hs
                                                     Wed angueure ( prodes year, prode so front, suf Lata) {
 typeday Smeet Mode (
                                                        Mode + nenemale = (reaterede (dola))
     ant data;
  Speed mode * next:
                                                          is ( " sear == prece) {
                                                              A year = A grown = new Mode:
  Hode )
  Mode + (seate Mode (sout data) {
                                                              elsef
      Mode * houseode = ( Mode * ) Malloc
                                                               (+ rear ) - next = new reade
                                                               A sear = new Modes
      (Some g woode ));
       numphodo => tata: tata;
        receased - rest = since !
                                                            rut dequeue ( Mode = 1 front ) (
                                                             2 ( * front == 12000) {
        refum necessode: surrenge - of eggs 1995
                                                               prints (" Ouece underflow ).
    word put (modera top, int date) &
      Mode + new Mode = (rate Mode (date))
                                                                return -2 :
        numerode - rest = # top;
                                                           plade & femp = x front ;
          t top crow Modey.
                                                            gut data etemp - data;
         weed pop ( Node ** top ) (
                                                             * food = (+ food) -> rost!
                                                               frue (temp):
           prints (" exac under flactor"):
                                                                return data:
             return - 1;
                                                                                          OUTPUT .
```

autaut. Steens operation; 30 popped from steere : 30 Popped from Stacio: 20 Glercies operations; Dequeue from queue; 1 00 Dequeue por que ; 200 MADELL BOOK A 2 TOOS capations a fusing on the x one x) Labour amor Z Coso A of expendence was of a state that a find a shot ( Comply with south worth

#### LAB PROGRAM 7:

WAP to Implement doubly link list with primitive operations

- a) Create a doubly linked list.
- b) Insert a new node at the beginning.
- c) Insert the node based on a specific location
- d) Insert a new node at the end.
- e) Display the contents of the list

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* prev;
  struct Node* next;
};
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if (newNode == NULL) {
    printf("Memory allocation failed.\n");
    exit(1);
  }
  newNode->data = data;
```

```
newNode->prev = NULL;
  newNode->next = NULL;
  return newNode;
}
void insertAtBeginning(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head != NULL) {
    (*head)->prev = newNode;
    newNode->next = *head;
  }
  *head = newNode;
  printf("%d inserted at the beginning.\n", data);
}
void insertAtEnd(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head == NULL) {
    *head = newNode;
    printf("%d inserted at the end.\n", data);
    return;
  }
  struct Node* temp = *head;
  while (temp->next != NULL) {
```

```
temp = temp->next;
  }
  temp->next = newNode;
  newNode->prev = temp;
  printf("%d inserted at the end.\n", data);
}
void insertAtPosition(struct Node** head, int data, int position) {
  if (position \leq 0) {
    printf("Invalid position.\n");
     return;
  }
  if (position == 1) {
    insertAtBeginning(head, data);
     return;
  }
  struct Node* newNode = createNode(data);
  struct Node* temp = *head;
  for (int i = 1; i < position - 1; i++) {
    if (temp == NULL) {
       printf("Position out of bounds.\n");
       free(newNode);
```

```
return;
    temp = temp->next;
  }
  if (temp == NULL) {
    printf("Position out of bounds.\n");
    free(newNode);
    return;
  }
  newNode->next = temp->next;
  newNode->prev = temp;
  if (temp->next != NULL) {
    temp->next->prev = newNode;
  }
  temp->next = newNode;
  printf("%d inserted at position %d.\n", data, position);
void displayList(struct Node* head) {
  if (head == NULL) {
    printf("The list is empty.\n");
    return;
```

}

```
}
  printf("List contents: ");
  struct Node* temp = head;
  while (temp != NULL) {
     printf("%d <-> ", temp->data);
     temp = temp->next;
  }
  printf("NULL\n");
}
int main() {
  struct Node* head = NULL;
  int choice, value, position;
  while (1) {
     printf("\nMenu:\n");
     printf("1. Insert at the beginning\n");
     printf("2. Insert at the end\n");
     printf("3. Insert at a specific position\n");
     printf("4. Display the list\n");
     printf("5. Exit\n");
     printf("Enter your choice: ");
     scanf("%d", &choice);
```

```
switch (choice) {
  case 1:
     printf("Enter value to insert at the beginning: ");
     scanf("%d", &value);
     insertAtBeginning(&head, value);
     break;
  case 2:
     printf("Enter value to insert at the end: ");
     scanf("%d", &value);
     insertAtEnd(&head, value);
     break;
  case 3:
     printf("Enter value to insert: ");
     scanf("%d", &value);
     printf("Enter position to insert: ");
     scanf("%d", &position);
     insertAtPosition(&head, value, position);
     break;
  case 4:
     displayList(head);
     break;
  case 5:
     printf("Exiting...\n");
```

```
exit(0);
default:
    printf("Invalid choice, please try again.\n");
}
return 0;
}
```

#### **OUTPUT:**

```
Menu:
1. Insert at the beginning
2. Insert at the end
3. Insert at a specific position
4. Display the list
5. Exit
Enter your choice: 1
Enter data to insert at the beginning: 45
Menu:

    Insert at the beginning
    Insert at the end

Insert at a specific position
Display the list
5. Exit
Enter your choice: 1
Enter data to insert at the beginning: 50
Menu:
1. Insert at the beginning
2. Insert at the end
Insert at a specific position
Display the list
5. Exit
Enter your choice: 3
Enter data to insert: 78
Enter position to insert at: 1
Menu:

    Insert at the beginning

Insert at the end
Insert at a specific position
4. Display the list
Exit
Enter your choice: 4
Doubly Linked List: 78 <-> 50 <-> 45 <-> NULL
Menu:

    Insert at the beginning

2. Insert at the end

    Insert at a specific position
    Display the list

5. Exit
Enter your choice: 2
Enter data to insert at the end: 34
Menu:

    Insert at the beginning

    Insert at the end
    Insert at a specific position
    Display the list

5. Exit
Enter your choice: 4
Doubly Linked List: 78 <-> 50 <-> 45 <-> 34 <-> NULL
```

```
ta) were programm to complement doubly linked
  lest with primetive operations
  0) (seate a doubly linked list
  b) Insert now node in doubly unted list
  () Insert the node bused on specific location
  d) Enjort anewnode at the end
 a) Display the contents & linked 1881
   #include & stdio.h)
   # mclade & stall b. hs
   Staut node of
      int data.
   Staut node * neat!
   struthodet provi
Street Mode + Create Node ( pot data)
    Stant Noder nene woder ( stant Moder ) malloc ( sive &
             (Speethode):
       new Node -, date = fater:
       New Mode - west: MULL;
     new Node -> pager = NULL;
         neturn new place.
Void mosest At Reginning Esteut noglet head, mit date X
  Strut mode new prode = coeate prode (data).
    Stored Model ferop - i head !
```

I ( hear 1: Nous) ( ho vord mest at End (stud productions, not down ( a head : newstocks Christmodes mandes Create Mode (data) Void ment as a grape parton (stoud Moders head, Strult Moderttemp = dut bead ; int date, not position) { if ( + head = = MULL) A head newelode Stanfrode \* Newwork: Create Mode (data) refusn: Street worken temp: \* hoad; habite (temponent) = Nucle) 8 (ps: = 2) { mosest say ( head idata); temp stemp - next temp - next new place. return! newpeader prev = temp! for (ortics isposing of temp]: HULL 144) 1088 display(start Mode" head) { Struct moder temp = bead; prints (" Doubly looked list "); fring ( postonis ) out of bounds ); unele (temp1: NULL) { se fann! printy ("Y.d )", tup -1 data). temp - demp - ment; New Mode - rest: temp- , next. new Node -) prove temp. Dunk In Durchi, T. of ( few por next = NOW) temp - marely -) polv = new Node: temps next = rew mode;

Mence: output 1. Ensert at the beginning d. Insert at the end WEDD. 3. Engest at the specific position placent atthe beginning a Intest at the end Lr. Display the less 3. Insert set specific position 5. Exest 9. Display the list Enter your choice : 4 5. 64B fates your choice: 2 Enter data tomsel at the bogging: 25 Menu: 1. Enjest at the beginning 2, Inject at the End 2. Engert at the beginning 3. Enjert at a specific position d. Engest cut the kind 2. Injest citthe specific position 4. Display ten lest 4. Display the seit 5. oust Enter your Choice \$ 5 5. ELEST Enteryour Cholce: 2 Enter data to insert at the end 52 Mene 1. Insert at the beginning d. Engest cet the potition end. Engest withe specific position" t Display the lest 5. exest Enter your children

### **LAB PROGRAM 8:**

Write a program

- a) To construct a binary Search tree.
- b) To traverse the tree using all the methods i.e., inorder, preorder and post order
- c) To display the elements in the tree

### **PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* left;
  struct Node* right;
};
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->left = newNode->right = NULL;
  return newNode;
}
struct Node* insert(struct Node* root, int data) {
  if (root == NULL) {
    return createNode(data);
```

```
if (data < root->data) {
     root->left = insert(root->left, data);
  } else {
     root->right = insert(root->right, data);
  }
  return root;
}
void inorder(struct Node* root) {
  if (root != NULL) {
     inorder(root->left);
     printf("%d ", root->data);
     inorder(root->right);
  }
}
void preorder(struct Node* root) {
  if (root != NULL) {
     printf("%d ", root->data);
     preorder(root->left);
     preorder(root->right);
}
void postorder(struct Node* root) {
  if (root != NULL) {
     postorder(root->left);
     postorder(root->right);
     printf("%d ", root->data);
int main() {
  struct Node* root = NULL;
```

```
int elements [] = \{1,2,3,4,5,6,7\};
int n = sizeof(elements) / sizeof(elements[0]);
for (int i = 0; i < n; i++) {
  root = insert(root, elements[i]);
}
printf("In-order Traversal: ");
inorder(root);
printf("\n");
printf("Pre-order Traversal: ");
preorder(root);
printf("\n");
printf("Post-order Traversal: ");
postorder(root);
printf("\n");
return 0;
```

### **OUPUT:**

```
In-order Traversal: 1 2 3 4 5 6 7
Pre-order Traversal: 1 2 3 4 5 6 7
Post-order Traversal: 7 6 5 4 3 2 1

Process returned 0 (0x0) execution time: 0.016 s
Press any key to continue.
```

```
80) write a programm
                                                              void invoider (Street moder soud) 1
a) to constitut a binary trusch true
 b) To convisi the tree asymptothe Methods i.e. in order
                                                                   (4 (800+1 - pucc) {
  precorder, pertorder, display all boursal lest
                                                                     10001421 (2001-) ph():
                                                                      printy (". Id", 1001 - idata)
 (a) booder with
                                                                       monde o ( soo - des inght)
  # anchede CHAIDAS
                                                                   void precorder (Stand Modes noot) {
  Ameleda Litalibhs
                                                                         18 (20011= harr)
  Street Mode &
                                                                       print ("11", not - data);
     and clasors
                                                                       precorder . ( NO 1 - left).
     Street red + left;
                                                                       precorder (2001 - right);
     Speed Moder 1: ght;
Street produce (seate productint data) { }
                                                                       the said of the
   Fruit Mode new Mode & estud Moder) malloc (like of (through)
     newhole - data - data;
      new reads - byt = new reads - right = Nucle;
                                                                 Voted postordes ( spul profex scot) (
                                                                      B (soot i - Noco) {
      defum heroglode.
Street Moder Insert ( Speed Moder rood, not don't )
                                                                       post order (2001 - lift)
                                                                        postando. (suot-> Ryhi);
     1 (2001=: NOTA)
                                                                        printy [" I'd" root -related );
       setura (sente Node (dates)
      (data Loop t - dala) {
          root -date lylement (root-) left, dates);
                                                                   tut mater () {
        4 elled
                                                                       Stant Modex 2001 = NULL,
            Just - mild = in(ed (2001 - right, data)
                                                                          mt element [] = (60; 30, 20, 40, 20, 60, 80, };
                                                                           Trid n = Six f (elements) I six of (elements (0)):
          offun root
                                                                           for (nd 1:0; 12 h; 1+1) {
```

```
good = wheat ( soot alements (1))
       pring ( To-order Taivessal : )
        Mostles (201);
        bisub ( | 10,1)
       being ( but-order lanesers ).
         [1 +000) apressed
         pring (" | n").
        prints ( post order Traversalis):
         postord-, (200+);
          psup (" | ~ ");
          Defuny 0;
 Output:
In-order -towersal: 20.30, 40, 50, 60, 70, 80
  pre-order toaveryal: 50, 30, 20, 40, 20, 60, 80
  post-order traversal: 20,40,30 60, 80, 20, 50,
```

## **LAB PROGRAM 9:**

- a) Write a program to traverse a graph using BFS method.
- 5 b) Write a program to check whether given graph is connected or not using DFS method

#### **PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
struct Queue {
  int items[MAX];
  int front;
  int rear;
};
void initQueue(struct Queue* q) {
  q->front = -1;
  q->rear = -1;
}
int isEmpty(struct Queue* q) {
  return q->front == -1;
}
void enqueue(struct Queue* q, int value) {
  if (q->rear == MAX - 1) {
    printf("Queue is full\n");
     return;
  if (q->front == -1) q->front = 0;
  q->rear++;
```

```
q->items[q->rear] = value;
}
int dequeue(struct Queue* q) {
  if (isEmpty(q)) {
     printf("Queue is empty\n");
     return -1;
  }
  int value = q->items[q->front];
  if(q->front == q->rear) {
     q->front = q->rear = -1;
  } else {
     q->front++;
  return value;
}
void BFS(int graph[MAX][MAX], int start, int n) {
  bool visited[MAX] = {false};
  struct Queue q;
  initQueue(&q);
  enqueue(&q, start);
  visited[start] = true;
  while (!isEmpty(&q)) {
     int node = dequeue(&q);
     printf("%d", node);
     for (int i = 0; i < n; i++) {
       if (graph[node][i] == 1 && !visited[i]) {
          enqueue(&q, i);
          visited[i] = true;
       }
    }
```

```
void DFS(int graph[MAX][MAX], int node, bool visited[MAX], int n) {
  visited[node] = true;
  printf("%d ", node);
  for (int i = 0; i < n; i++) {
     if (graph[node][i] == 1 \&\& !visited[i]) {
        DFS(graph, i, visited, n);
     }
   }
int main() {
  int n = 6;
  int graph[MAX][MAX] = {
     \{0, 1, 1, 0, 0, 0\},\
     \{1, 0, 1, 1, 0, 0\},\
     \{1, 1, 0, 1, 0, 0\},\
     \{0, 1, 1, 0, 1, 1\},\
     \{0, 0, 0, 1, 0, 1\},\
     \{0, 0, 0, 1, 1, 0\}
  };
  printf("BFS starting from node 0: ");
  BFS(graph, 0, n);
  printf("\n");
  bool visited[MAX] = {false};
  printf("DFS starting from node 0: ");
  DFS(graph, 0, visited, n);
  printf("\n");
  return 0;
```

# **OUTPUT:**

```
BFS starting from node 0: 0 1 2 3 4 5
DFS starting from node 0: 0 1 2 3 4 5

Process returned 0 (0x0) execution time : 0.000 s

Press any key to continue.
```

de l'ample broudent to previde adocht (b) betwite a progress to tours se a graph using DES yetho #melude 251 A.D. hs Amelode & Halibhy Harchede KAdbiol. h Aleque Max 100 Street queuch oul items [minx] And front; nd seas; Void me queue (Stored process of) { go trail = ?; and is Empty (Street Queces 9) ( Deturneys from == -3 Noted Enquere (Smul matte ouere & q, no value) of A (d- ) reas = = +100 - - 5) { printy (" pueces is full fa"). refunn; 18 79-1 pool == -2 19-1 Food =0. 9-1 reas +1) geritems (germen) - value, out dequere (Speel Prece og ) !

```
8 (11 compty (9)) f
  purch ( Laure 12 Ember / 1, 1);
  Defann-3;
orfiner: 9- item (9) font ];
  8 (9-1+xx1==9-1 year) (
    9-1 port = 9-1 (01) -- 2:
 ) CHEL
     9-> [rad +4)
   return value,
 road sis (not graph (MAR) (MAK); and stead; ont ),
  bool risited [MAX] = { false};
     Fruit piece 9;
      retail que ( Eg);
    enqueux ( Eq. Acat)!
     Visited ( Shoot ): true;
  welle (! 11 Empty ( 891) {
      out mode : dequar ( Eq ).
     prints ("ind", node);
    for (ord: 50; 111; 141) {
      & graph ( node ] ( ) ] = 1 { } | visited () ){
```

```
enque ( 49, 1):
      Visited (1) - + me
vold per (mt gruph (max) (max), not node, bool visited
         (Max), men) {
    Visited (node) tree;
     prints (" rd", node);
    for (mt ? = 0; in ; 14) (
      1 (graph (node][:]=1 { [ visited[:]]
         DES (golph, i, visited, n)
   helmom () (
      mtn = 6
      mt grouph (MA) [ MA) = {
         h 0, 2, 2, 0, 0, 09,
         12,0,2,2,0,0%
         (2,2,0,2,0,0,}
        (0, 2, 2, 0, -2, 2)
         40,0,0,1,0,35,
         10,0,0,2,2,0}
```

prints ("Bre Stanting from node 0:"): BFS(gociph 0, n); bool visited [mna] = { faire}: Pring ("DES Storting from node 0: "); DES (gouph 10, visited, u) prints (" [""); seturn o Output: BFS (Hordring from hode 0: 0,1,2,3,4,5 DES Starting from node 0: 0 2 23 4 5

