VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT on

Data Structures using C Lab

(23CS3PCDST)

Submitted by

SUJAN G E(1BM23CS347)

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 Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled "Data Structures using C Lab (23CS3PCDST)" carried out by SUJAN G E (1BM23CS347), who is bonafide student of B.M.S.College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering inComputer Science and Engineering of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of Data Structures using C Lab (23CS3PCDST) work prescribed for the said degree.

Namratha M	Dr. Jyothi S Nayak Professor &HOD Department of CSE, BMSCE
Assistant Professor Department of CSE, BMSCE	

Index

Sl. No.	Date	Experiment Title	Page No.
1	30/09/2024	Stack Implementation using arrays	1-6
2	07/10/2024	Infix to Postfix Conversion	7-12
3	15/10/2024	Queue implementation using arrays	13-23
4	21/10/2024	Circular Queue implementation using arrays	24-31
5	29/10/2024	Insertion Operation in Singly Linked List & Leet Code Valid Parenthesis	31-40
6	11/11/2024	Deletion Operation in Singly Linked List & Leet Code Daily Temperature	41-56
7	02/12/2024	(i) Multiple Operation in Singly Linked List(ii) Stack and Queue in Singly Linked List	57-66
8	09/12/2024	Insertion operation in Doubly Linked List	66-72
9	23/12/2024	Binary Search Tree Insertion with pre-order,in-order and post-order traversal and display.	73-76
	23/12/2024	BFS and DFS traversal in Graph using adjacency matrix	77-81

Github Link:

https://github.com/Sujan279/DS

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

```
# Include 21th 5tack Operation: 3017124
#Enclude 25tdio.h>
  #include 21 fdbool. hs
  Struct Stack
   Ent an [20];
                               Noted fall stack ( stack & sp) and size)
   ant top === ;
   Ent 3934;
  3;
                                      Sp >51 7 -5178,
  bool Examply (Shoul stack * SP)
 d of (Sp -> top== -1)€
  & return true;
  return false;
 bool is Full (Struct stack * Sp)
d of (Sp->+op== Sp-> Si38}-1)
 d return true;
  seturn falle;
void push (struct stack * $ p, int val)
  & [i is full (3P))
  sp > am (top) = val;
  sp -> top ++;
ysp -> arr [sp-> top] = val;
 else a print ( "stack overflow)", y
End pop (struct stack & Sp)
 def (!is empty (SP))
 Ent date = Sp -> arr[fop];
     Sp -> top --;
     odurn dala;
   esse (")tack underflow");
```

```
Int peek (Struct Stack * sp)
   return sp -> arr [top] [sp-> top];
void traverse (struct stack + Sp)
  too (Ent P=Sp -> size-1; ?>=0; i --).
  Print[ (".1. dln", sp > on(i));
void mainers
   Struct stack * Sp;
   Sp = (Struct stack) mallac (six six of (struct stack));
    pash (sp,1);
    purh (Sp 12);
    pop (5p);
    peek (sp);
    traverse (sp);
  GUTPUT:
     popped value : 2
     Top value : 1
      Stack elmosts: 1
OUTPUT!
    2->3
     1-51
    operation.
     POP(N;
     pop(vi,
     pop(v); under flow > Peek (v) > 0/Pupty stack
     push (v,30);
     push (V, 20);
     pash (v, ro);
    Parh (V, Gd), -> o vertow
     ROOKIV) + -> 60
```

CODE:

```
#include <stdio.h>
#define MAX 5
int stack[MAX];
int top = -1;
void push(int element) {
  if (top == MAX - 1) {
    printf("Stack Overflow! Unable to push %d\n", element);
  } else {
    stack[++top] = element;
    printf("Pushed: %d\n", element);
}
void pop() {
  if (top == -1) {
    printf("Stack Underflow! The stack is empty.\n");
  } else {
    printf("Popped: %d\n", stack[top--]);
}
void display() {
  if (top == -1) {
    printf("The stack is empty.\n");
  } else {
    printf("Stack elements are: ");
    for (int i = top; i >= 0; i--) {
       printf("%d", stack[i]);
    printf("\n");
}
int main() {
  int choice, element;
```

```
while (1) {
  printf("\n--- Stack Operations ---\n");
  printf("1. Push\n");
  printf("2. Pop\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 push: ");
       scanf("%d", &element);
       push(element);
       break;
     case 2:
       pop();
       break;
     case 3:
       display();
       break;
     case 4:
       printf("Exiting program.\n");
       return 0;
     default:
       printf("Invalid choice! Please try again.\n");
  }
```

```
Enter the limit of the stack: 5

Stack Operations:

1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 1
Pushed 1 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 2
Pushed 2 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 2
Pushed 2 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter your choice: 1
Enter value to push: 4
Pushed 4 onto the stack.
```

```
Stack Operations:

1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 5 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 4
Stack elements: 4 3 2 1

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 4

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 4 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 3 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 3 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 3 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 3 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 2 from the stack.
```

WAP 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)

```
Infinto Postfex.
# Endudez Sideo. h>
# Enclude < math. h>
# Enclude < Stallb.h>
Ent (Sop (charc)
      return 3;
   else ? (c == 1/1 11 c == 1 x)
      return 2º
  else 9 (c==++1 11 c== '-1)
  else
     return -1; 7
char ass (charc)
      return R:
vold entertoposties (const chor + s)
   Ent len = strlen(5);
  charroses = (charro) malloc (lan+i);
   char* stack = (char*) malloc (lan);
   Put result India = 0;
  and stack Indes = -1;
```

```
Stack Endex + +;
       Stack [++ stack Index
if ( ires 11! stack)
   Print[ " Not Possible ")
   return;
for (8=0; 82 len; 8++)
 char c = 5[1]:
 8 ((c >= 'A' && C X='Z') 11(c>='a' && C (c='z') 11(c>='0' x2 (x='qi))
      res [resIndus++] = c;
else 9 (c== (?)
   stack [++ stack Indus] = c;
else if (c == 1))
   While (stack Indus) = 088 stack [stack Indus] 1 = '(') }
        res [resIndox++] = stack[stackIndex-];
      stackthden -- "
de
    while (stacktydes >=0 && (Gopco) & Trop (stack[stackIndus])
     Il (isop(c) == isop(stack[stack[ndex]) && assec) ==(L)))
       restresIndex++] = steeck[stackIndex--];
        Stack[++ Stack_Index] = C;
     3
```

```
while (stack studen > = 0)
       res[restudent+] = Stack [stacktuden-];
   resultiresulterdow = 101;
      Parall"+51", 2050;
    tree (res);
     bree (stocke);
    Put mount
     a choo expt] = "a+b*(c^d-e^(+g*h)-9";
(a+b*c)
(a+b*c);
     returno;
  3
OUTPUL :
    a+6* (c^d-e)^(6+g*h)-9
    Postfix expression: about - tgh + 1 x + 4 -
       0
                                     Maruel
  a+6*C
                               atbee
  Postfix expression: about
                                        a+6C*
                                 a6c-x+
```

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int precedence(char c) {
  if (c == '^{\prime}) return 3;
  else if (c == '*' || c == '/') return 2;
  else if (c == '+' \parallel c == '-') return 1;
  else return -1;
}
char associativity(char c) {
  if (c == '^') return 'R'; // Right-to-left associativity
  return 'L'; // Left-to-right associativity
}
void infixToPostfix(const char *expr) {
  int len = strlen(expr);
     char *result = (char *)malloc(len + 1);
  char *stack = (char *)malloc(len);
  int resultIndex = 0;
  int stackIndex = -1;
  if (!result || !stack) {
     printf("Memory allocation failed\n");
     return;
   }
  for (int i = 0; i < len; i++) {
     char c = \exp[i];
         if ((c \ge 'a' \&\& c \le 'z') || (c \ge 'A' \&\& c \le 'Z')) 
        result[resultIndex++] = c;
         else if (c == '(') {
        stack[++stackIndex] = c;
```

```
else if (c == ')') {
       while (stackIndex \geq 0 && stack[stackIndex] != '(') {
          result[resultIndex++] = stack[stackIndex--];
       stackIndex--; // Pop the '(' from the stack
          else {
       while (\text{stackIndex}) >= 0 \&\& \text{precedence}(c) <= \text{precedence}(\text{stackIndex})) 
          if (precedence(c) == precedence(stack[stackIndex]) && associativity(c) ==
'R') break;
          result[resultIndex++] = stack[stackIndex--];
       stack[++stackIndex] = c;
  }
  while (stackIndex \geq 0) {
     result[resultIndex++] = stack[stackIndex--];
  }
  result[resultIndex] = \\0'; // Null-terminate the result
  printf("Postfix expression: %s\n", result);
  free(result);
  free(stack);
}
int main() {
  char expr[] = "a+b*(c^d-e)^(f+g*h)-i";
  infixToPostfix(expr);
  return 0;
}
```

```
Output

Postfix expression: abcd^e-fgh*+^*+i-

=== Code Execution Successful ===
```

a) 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

```
# Enclude < stdio. h >
                  # include cstdlib-hs
Enqueu (x)
  if (ISFULL)
 Prent (" queue is Full");
else if ( Is empty ())
  3
 MEE
  If (Isempty ())
   Print (" que is Empty")
else
d front = front +1;
```

```
Is Full ()
       ( if (rear = = 5126 -1)
           return True;
          return False;
     Is empty ()
        if ( front == -1 & { secor == -1)
void Display ()
    1/ (is compty ())
     Prosalf (« Queu à empty ");
    print[(" Que elements ");
     for (i=0; icreas; i+) ("a) ("a)
      2 Print ("1.d", A[1]);
      Pront((In);
```

```
#include <stdio.h>
#define SIZE 5
int queue[SIZE];
int front = -1, rear = -1;
void insert(int value) {
  if (rear == SIZE - 1) {
     printf("Queue Overflow\n");
     return;
  if (front == -1) front = 0;
  queue[++rear] = value;
void delete() {
  if (front == -1 \parallel front > rear) {
     printf("Queue Underflow\n");
     return;
  front++;
void display() {
  if (front == -1 \parallel front > rear) {
     printf("Queue is Empty\n");
     return;
  for (int i = front; i \le rear; i++) {
     printf("%d ", queue[i]);
  printf("\n");
int main() {
  insert(10);
  insert(20);
  insert(30);
  display();
  delete();
```

```
display();
  delete();
  delete();
  delete();
  return 0;
}
```

```
Output

10 20 30
20 30
Queue Underflow

=== Code Execution Successful ===
```

b) WAP to simulate the working of a circular 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

```
WEER - 3
# andude 2stdig. h>
the raduale a std lib. hs
 Put Bront;
Put esfull (struct +a)
 int isempty (struct va)
     return ( q = 6 = = q > 7);
   3
void etrque (smult quin » or, int val)
    Proto(° Overflow");
exit(o);
   else
```

```
Put deque (struct qui so ex)
  of (rsempty(a))

Printf(" underflow");

exit (0);
 * inta=a>orrta->front;
  returna;
vold display (struct queuxar)
of (15 empty car)
 t for li=a>pont; ic=ay-reus; i+t)
    Proff olemet red "syd", ", arantil);
d Proint[1" underflows");
 exit (0);
```

```
int main ()
 struct quine q; mt ch;
 Q-size = 10;
 9- front =-1
Q. reer = -1;
Q-arr = (into) malloc (q-size x size of (int));
Proof [" Enter cheice");
Print [ " 1. Intre or at rear 2. Delete at front 3. Displaying;
Scanf ( 4.0 ", & ch);
Switch (ch)
 of
    care 1 . Posint/ (" onto element")
             scenf (" . d", & val);
              enquine (&ay, val);
   to break; but but the stands of the
    Cene 2: deque (801);
            Protol" poped vd ", Legue (80);
   care 3: d'splay (a);
brent; (" bool falon 4) plus 9
  defaillt. Print ["Ivaled input");
```

output MENU; 1. Enqueu 2 - Dequeue 3. Display 4. Oxit Enter choice : 1 enter element: 10 entre choice: 1 Onter element : 20 Enter charce , 1 enter element: 30 Enter charact: 401 enter clement 340

```
#include <stdio.h>
#define MAX 5
int queue[MAX];
int front = -1, rear = -1;
void insert(int value) {
  if ((front == 0 \&\& rear == MAX - 1) || (rear == (front - 1) % (MAX - 1))) {
    printf("Queue Overflow\n");
    return;
  }
  if (front == -1) {
    front = rear = 0;
  } else if (rear == MAX - 1 && front != 0) {
    rear = 0;
  } else {
    rear++;
  queue[rear] = value;
void delete() {
  if (front == -1) {
    printf("Queue Underflow\n");
    return;
  if (front == rear) {
    front = rear = -1;
  } else if (front == MAX - 1) {
    front = 0;
  } else {
    front++;
}
void display() {
  if (front == -1) {
    printf("Queue is Empty\n");
    return;
```

```
if (rear >= front) {
     for (int i = front; i \le rear; i++) {
       printf("%d ", queue[i]);
     }
  } else {
     for (int i = front; i < MAX; i++) {
       printf("%d ", queue[i]);
     for (int i = 0; i \le rear; i++) {
       printf("%d ", queue[i]);
     }
  printf("\n");
int main() {
  int choice, value;
  do {
     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:
          break;
       default:
          printf("Invalid choice\n");
  \} while (choice != 4);
  return 0;
```

output:

```
Clear
  Output
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 23
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 34
1. Insert
2. Delete
3. Display
4. Exit
Output
                                                                 Clear
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 34
1. Insert
2. Delete
Display
4. Exit
Enter your choice: 4
=== Code Execution Successful ===
```

WAP 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

```
@ write a program to Implement
                                     Olngly Unked West with following
     operations.
    · Create a linked lest
   · Insertion of a node at first position & at end of list.
    Display the contents of the Unked CEst.
    # Enclude = stdio. h >
    # Endude < stalib h>
    struck Node
    a gent data;
      Struct Noon * next;
    8.
   Struct Node & create Node (Ent duta)
    Struct Node = new Node = (Struct Node +) malloc (Size of Struct Node);
   new Node > data = data;
   new. Node - next = Null;
   return newwode;
  Struct Node & Create Lanked lest lent deutoArray [] Port Size)
  & struct woods
#Enclude < stdlo.h>
tendude < stall b.h >
 Struct Noch
 Ent data:
 node + next .
truct mode & createnocle (Ent data)
struct Node * new Node = (struct Noch *) mallac (size of (struct Node));
newNode -> data =data;
necowode - next = NULL;
return neconodi,
```

```
vord theret (noch * head, int data)
 program to Implement asody unked Wast wath following
    node * new = Create (data);
    ? ( to head == Null)
      + head = new node;
    else
       Struct woch + temp = +head=
       while (temp-snext) = wull)
       1 temp = temp > next;
        temp -> next= new Node;
void Ensentat Bog [Struct wode on head, Int date)
d struct Node new node = ( reale Node (data)
   newworld = snext = x head; 3) inthonal than a world have
   whead = new Node;
vold display (struct node + head)
 struct node = temp = head;
  mype ftenb : = NOM)
   Print[" 1-d", tap -> data),
         shud wich * newworld = (shut nous ") malter (size of
 2
```

```
struct woods scheard = Null;
Ent ch, dada;
utile (frue)
Printflu toto choice voinsent extreg 2. Ensent at end 3. display, 4 Expit!);
 Scanglo 1-d", 8 ch);
   switch (ch)
     care: printfluents data"; scanflu 1.d", &data);
            Prisent Al beg(& nead, douba);
             Pareous.
      cau 2: Print[[" tille decla");
scan["1-d", & decla");
               Prosent and (& head, doda);
              break;
      cures: display [fread);
                break.
     can 4: expt(0);
     default: print[ "Invalod");
     returno:
```

```
OUTPUT!
 Choose an option:
 s. Insert at end?
 2. Insert at bega
 3. Display
 4 . Gost
enter your chorce: 1
 Enter data : 10
enter your choice 32
 Onto data: 20
 enter your choice: 2
enter data: 33
enter your choice: 3 (halob man in flored ? co sum)
current lest: 33 20 20
enter your choice: 4 hours 1 has been
exot
```

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data:
  struct Node* next;
};
struct Node* head = NULL;
void createList(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  if (head == NULL) {
    head = newNode;
  } else {
    struct Node* temp = head;
    while (temp->next != NULL) {
       temp = temp->next;
    temp->next = newNode;
  }
}
void insertFirst(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = head;
  head = newNode;
}
void insertAtPosition(int data, int position) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  if (position == 1) {
    newNode->next = head;
    head = newNode;
    return;
  }
```

```
struct Node* temp = head;
  for (int i = 1; temp != NULL && i < position - 1; i++) {
    temp = temp->next;
  if (temp == NULL) {
    printf("Position out of range\n");
    return;
  newNode->next = temp->next;
  temp->next = newNode;
}
void insertLast(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  if (head == NULL) {
    head = newNode;
  } else {
    struct Node* temp = head;
    while (temp->next != NULL) {
       temp = temp->next;
    temp->next = newNode;
}
void displayList() {
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d", temp->data);
    temp = temp->next;
  printf("\n");
int main() {
  int choice, data, position;
  while (1) {
```

```
printf("1. Create List\n");
printf("2. Insert at First Position\n");
printf("3. Insert at Any Position\n");
printf("4. Insert at Last Position\n");
printf("5. Display List\n");
printf("6. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
  case 1:
     printf("Enter data to create list: ");
     scanf("%d", &data);
     createList(data);
     break:
  case 2:
     printf("Enter data to insert at first position: ");
     scanf("%d", &data);
     insertFirst(data);
     break:
  case 3:
     printf("Enter data and position to insert: ");
     scanf("%d %d", &data, &position);
     insertAtPosition(data, position);
     break;
  case 4:
     printf("Enter data to insert at last position: ");
     scanf("%d", &data);
     insertLast(data);
     break:
  case 5:
     displayList();
     break;
  case 6:
     exit(0);
  default:
     printf("Invalid choice\n");
}
```

```
Initial linked list:
1 -> 2 -> 3 -> NULL
Menu:
1 for addfirst
2 for addLast
3 to display
Enter your choice 1
enter the ele to add 0
Menu:
1 for addfirst
2 for addLast
3 to display
4 to exit
Enter your choice 3
0 -> 1 -> 2 -> 3 -> NULL
Menu:
1 for addfirst
2 for addLast
3 to display
4 to exit
Enter your choice 2
enter the ele to add 4
Menu:
1 for addfirst
2 for addLast
3 to display
4 to exit
Enter your choice 3
0 -> 1 -> 2 -> 3 -> 4 -> NULL
Menu:
1 for addfirst
2 for addLast
3 to display
4 to exit
Enter your choice
```

WAP 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.

```
WAP to Emplement singly kinked list with
  operations.
@ Creale a Winked CAL
6 Delition of first element, specified element & last element
@ Display the contents of the winked list.
us # Enclude 2 staio. h>
  # Enclude < stdlib. h>
  Struct Noded
    Ent data;
     struct node * next;
  Struct Node + head = NULL ;
  void create Centred Cest (ent elements [], Ent n)
      for (Ent 8=0 ; Pan , ; 9++)
         ent data = eliments [:7:
         struct Node * new node = (struct node *) malloc (single) (struct
         newnode -> dala = dala;
         newnode > next = Null;
       of (head == NULL)
        d head = new rode;
        8
      elsez
      Struct Node temp = head;
      while (femp :> next! = NULL)
       temp = temp -> next;
    femp > next = new node;
 3
```

```
vota deletiferst ()
     Knowed let will fellowing
      of (node == null)
         Prentf(" let 9's empty");
         return;
       3
      Struct Node temp= head;
      head = head -s next;
     free (temp)
     Prenty | " first element deleted")
                                     void delle eleud struct works houd in
    yord delete element (ent valeus)
                                     dstruct node sop = head;
                                                  *ay-head;
       of (nead == NULL)
       a print[[" bit & comptey");
                                      struct rock
                                          whole ( & pos)
    if (head > data = = value)
     of of the
       struct node * timp = head;
       head = head -> next;
       free (temp);
       Print[[" element r. d deleted, values);
       return;
Struct wools * temp = head;
while ( temp -> next != NULL && temp-> next -> data! = values) =
     temp = feup -> next ",
 3 Pb (temp - next! = NULL)
    Struct Noch to Delete = temp->next;
    temp -> next = temp -> next -> next;
    free (tobelete);
    Prontf (" Element 1. d Deleted Nature);
   Probable " claimet rid not found ", value);
```

```
vold ddetlast()
     2 & (head == NULL)
       Prantflourst is empty, com't deletic);
       return ;
     g ( head -> next = = NUX)
       d free ( heard);
       head = NULL;
       Exent( " hard element deleted ");
       return;
ites) Street woods * temp = head
    while (temp -> next! = NULL & & temp -> next -> heat! = NULL)
      d temp=temp=next;
      gree (temp > next)
     temp -> next = WULL ;
    Pronto (" Last element delebed");
    8
  word daplaylist()
   d of (head == NULL)
      prentila Let 12 compta a);
      return,
 Struct rode & temp = head;
  while (temp!=NULL)
   d protoff ("1.d -> ; temp -> data);
     temp = temp - s next;
   Point ( NULL 11);
  8
```

```
Ent magne)
I get ch, value, h;
   ent elements [10];
  while (1)
     Prentfl"1. Create (Enked Cest");
    Prends " 2. Delete the ferst element");
    Prints 3. Delete prosticulos diment");
    Profife 1. Delile last element");
    Pointfl' 5. Display the Conted Ust);
    Print[1" 6. exet");
    Proceed the your choice")
    Schanf[" 4 d", &ch);
d case 2: Print[ 6 feter the note of elect you want to involt;
switch (ch)
          Scar [ " 1.d" &n);
          Prosell' trale the elevents");
           foolinties; een; it)
            d scuff, 18, 18 offent Ci);
            Create Catalitat ( deut, n);
           Probably hinked but created ")
           break,
          delete Forsti),
 Care 2 :
          breaks,
         Dorsy o Brown ster postatoon to delet.")
care 3"
           scan) 1 7 d' , &d);
           deletelent (head, d);
            break;
```

case 4 3 dellettent (); break, display with; caus . breaks care 6 : Drive [1" garate Goit ")", exitio) Obfault : Postfi Envalled () Locutmo; OUTPUT: MENU: 1. Create a linked Oit 20 polale the first element 3. Delete then element by Position

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data:
  struct Node* next;
};
struct Node* head = NULL;
void createList(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  if (head == NULL) {
    head = newNode;
  } else {
    struct Node* temp = head;
    while (temp->next != NULL) {
       temp = temp->next;
    temp->next = newNode;
}
void deleteFirst() {
  if (head != NULL) {
    struct Node* temp = head;
    head = head->next;
    free(temp);
}
void deleteLast() {
  if (head != NULL) {
    if (head->next == NULL) {
       free(head);
      head = NULL;
     } else {
       struct Node* temp = head;
```

```
while (temp->next != NULL && temp->next->next != NULL) {
         temp = temp->next;
       free(temp->next);
       temp->next = NULL;
  }
void deleteSpecified(int value) {
  if (head != NULL) {
    if (head->data == value) {
       struct Node* temp = head;
       head = head->next;
       free(temp);
     } else {
       struct Node* temp = head;
       while (temp->next != NULL && temp->next->data != value) {
         temp = temp->next;
       if (temp->next != NULL) {
         struct Node* toDelete = temp->next;
         temp->next = temp->next->next;
         free(toDelete);
    }
void displayList() {
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d", temp->data);
    temp = temp->next;
  printf("\n");
int main() {
  int choice, data, value;
```

```
while (1) {
  printf("1. Create List\n");
  printf("2. Delete First Element\n");
  printf("3. Delete Last Element\n");
  printf("4. Delete Specified Element\n");
  printf("5. Display List\n");
  printf("6. Exit\n");
  printf("Enter your choice: ");
  scanf("%d", &choice);
  switch (choice) {
     case 1:
       printf("Enter data to create list: ");
       scanf("%d", &data);
       createList(data);
       break:
     case 2:
       deleteFirst();
       printf("First element deleted.\n");
       break;
     case 3:
       deleteLast();
       printf("Last element deleted.\n");
       break;
     case 4:
       printf("Enter value to delete: ");
       scanf("%d", &value);
       deleteSpecified(value);
       printf("Specified element deleted.\n");
       break:
     case 5:
       displayList();
       break:
     case 6:
       exit(0);
     default:
       printf("Invalid choice\n");
  }
}
```

```
Initial linked list:
1 -> 2 -> 3 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 1
After deleting the first node:
2 -> 3 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter vour choice: 3
Enter the value to be deleted: 3
After deleting the specified node:
2 -> 4 -> 5 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 2
After deleting the last node:
2 -> 4 -> NULL
Menu
1. Delete first node
2. Delete last node
3. Delete specified node
4. Display list
5. Exit
Enter your choice: 4
2 -> 4 -> NULL
```

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

```
60 With a program to employeed aloge that It's t
 With following operation, sost the linked lost, Revene
 the Unked lot, concederation of two anked lists.
n) #Enclude estatio. 6>
  # Poclude 2 8686.6>
  Struct rode of
     Int data;
     struct rode + next;
  Stouck node * next;
 & Struct nade > Create linked (Est () 1
   Struct node & head = NOLL;
   shud node temp = NULL',
   struct node * newnode;
   ind n, value;
  Prentf[" Number of elements");
  scan ( " Id", 21);
  tos ( Put ?=0; ?Ln; ?++)
   Print( " ento");
   Scanf 1" xd', &value");
   new wode = (struct node ») malloc (58 ze of (5 truct node));
   newnode -> data = value;
   newworld & next = NULLS
  f ( head == NULL)
    3 else
      temp = newNode;
    return head;
```

```
vold gort lented let (5 hut Node * * nead) 1 = quet a short burte
      Eff + head == NULL) returns, (July = 1 Jan - grant ) allow
      struct node arread = & head;
      Struct Node & Endex - NULLS,
      gut Lemp;
      while ( current ! = NULL)
       Endex = current -> next;
       whell ( Poden ! = NULL) of
         8 (current -> data > Endex -> data)
            temp = current -stata;
             current -> data = Produc -> data
            Ender -> data = tempo;
           Endex - Prolio - s next;
          current = current -> heat;
     2
 vold reverse (entred (if (struct node so head)
  & struct node & prev = NULL;
     * Current = * head ;
     A next = NUCLO,
     while ( correct ! = NULL)
      d prot = current -> next;
       current > next = Prev",
        Prev - current ;
        current - next;
   * head - Arevo
Struct node corralmabilist (Struct moder head). Struct head & head
Pf (head 1 = NULL) return head 2;
of (head == NUL) roction heads;
```

```
Struct node + temp = heads;
    while ( fremp -> next! = NULL)
      I temp = temp -> next
     temp-> next = head 2;
     return heard 1;
vold Pointist (5 trut nade & head)
 d street node * tump = head;
   while (tamp! = NULL)
     Avent["1-d", temp->dota);
    temp = temp -> next;
   Pront] [" (m);
  & smuch noclet Ust = create (Poked (Pot C);
Ent maln ()
   struct node × CB+2 = Creatilented Lafe);
  Poerty (" List 1", ")",
   Prentlest (Wit1);
   Poent ( lest 2 5 1);
    Portlet (Ust2);
    Sortlinked list (& list1);
    Pronty (" stored Ost 1:");
     Print list (1st 1);
     reverse linted list (& lists);
     Poentf 1 Reversed (1st 2:11);
     Poenelist (lists);
    Struct node * concatenated lest = concatenatilist (list 1, lest 2);
    Prend(" con calinated (at:");
   Proceeding ( Concaterated (Bt);
    netraren 0;
```

OUTPUT! Number of elevents : 4 Enter value : 3 Enter value ? 2 Enter value: 4 erder valu: 8 Number of elements: 4 Enter value : 4 enter value : 6 enter value: 7 onter value : 2 List 1: 3 2 4 8 ASS + 9: 4 6 7 2 sorted Osts: 2 3 48 Reversed Ristz: 843 2 Concatinated 13t: 8 4 3 2 4 6 7 2 Struct noon * Party:

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* next;
};
struct Node* head = NULL;
struct Node* head2 = NULL;
void createList(struct Node** head, int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  if (*head == NULL) {
    *head = newNode;
  } else {
    struct Node* temp = *head;
    while (temp->next != NULL) {
       temp = temp->next;
    temp->next = newNode;
  }
}
void displayList(struct Node* head) {
  struct Node* temp = head;
  while (temp != NULL) {
    printf("%d", temp->data);
    temp = temp->next;
  printf("\n");
void sortList(struct Node* head) {
  struct Node *i, *j;
  int temp;
  for (i = head; i != NULL; i = i->next) {
```

```
for (j = i - next; j != NULL; j = j - next) {
       if (i->data > j->data) {
         temp = i->data;
         i->data = j->data;
         i->data = temp;
    }
void reverseList(struct Node** head) {
  struct Node *prev = NULL, *current = *head, *next = NULL;
  while (current != NULL) {
    next = current->next;
    current->next = prev;
    prev = current;
    current = next;
  *head = prev;
void concatenateLists(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() {
  int choice, data;
  while (1) {
    printf("1. Create List 1\n");
    printf("2. Create List 2\n");
    printf("3. Display List 1\n");
```

```
printf("4. Display List 2\n");
printf("5. Sort List 1\n");
printf("6. Reverse List 1\n");
printf("7. Concatenate Lists\n");
printf("8. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
  case 1:
     printf("Enter data to create List 1: ");
     scanf("%d", &data);
     createList(&head, data);
     break;
  case 2:
     printf("Enter data to create List 2: ");
     scanf("%d", &data);
     createList(&head2, data);
     break:
  case 3:
     printf("List 1: ");
     displayList(head);
     break;
  case 4:
     printf("List 2: ");
     displayList(head2);
     break;
  case 5:
     sortList(head);
     printf("List 1 sorted.\n");
     break:
  case 6:
     reverseList(&head);
     printf("List 1 reversed.\n");
     break;
  case 7:
     concatenateLists(head, head2);
     printf("Lists concatenated.\n");
     break:
  case 8:
```

```
exit(0);
    default:
        printf("Invalid choice\n");
    }
}
return 0;
}
```

```
List 1: 5 -> 1 -> 9 -> 3 -> NULL
List 2: 8 -> 2 -> 4 -> NULL

List 1 after sorting: 1 -> 3 -> 5 -> 9 -> NULL

List 2 after reversing: 4 -> 2 -> 8 -> NULL

List 1 after concatenation with List 2: 1 -> 3 -> 5 -> 9 -> 4 -> 2 -> 8 -> NULL
```

b) WAP to Implement Single Link List to simulate Stack & Queue Operations

```
GO WAP to amplement sample bank let to semulat
  stack & queu operations.
#Enduele Lotalso. hs
+ Enclude < stollib.h>
 Struct Node &
       Ent dala:
       struct wool & Rept;
  35
Smict Node * CrookNode (Ext data)
  shuck node + new wode = (shuck node ) mallac (seno b[shuck woods);
   of (new wood == NULL)
      a partille memory allowation fallied"),
        enotte);
      newNool -> data = data;
       reamode -> next = NULL;
       return new node;
vold pur (smuet node votop, int data)
    2 struct wood & newwood - create mode (docta);
       newwoode & new t= top;
        +lop = new Node,
Ent pop (struct Node xxtop)

d if (xtop == NULL)

Raete an-1;
      3
```

```
Node & tup = & front;
    int dequed oute = temp s duta;
    - front = ( to front) - snext;
     if ( to front = = NULL)
         * rear = NULL;
     tree ( tup);
     return dequet Octa;
 int front ( wode so front )
    return front = NULL;
Ent main ()
 Node + sp = null;
 Node # OF = NOLL;
 indicent Cook & a front now illen - 20 * non
Pruty La reacted); when I man grown a modern a modern
parh (885T, 10);
purh (85T, 20);
                      suredd) - neck - neumock :
                               · spound - L =
   Ent deguer Conact nock on poort, short nock now
                       mother a oriflown);
```

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data:
  struct Node* next;
};
struct Node* stackTop = NULL;
struct Node* queueFront = NULL;
struct Node* queueRear = NULL;
// Stack Operations
void push(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = stackTop;
  stackTop = newNode;
}
int pop() {
  if (stackTop == NULL) {
    printf("Stack is empty.\n");
    return -1;
  struct Node* temp = stackTop;
  int data = temp->data;
  stackTop = stackTop->next;
  free(temp);
  return data;
}
void displayStack() {
  struct Node* temp = stackTop;
  if (temp == NULL) {
    printf("Stack is empty.\n");
  } else {
    printf("Stack: ");
    while (temp != NULL) {
       printf("%d", temp->data);
```

```
temp = temp->next;
    printf("\n");
// Queue Operations
void enqueue(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->next = NULL;
  if (queueRear == NULL) {
    queueFront = queueRear = newNode;
    return;
  queueRear->next = newNode;
  queueRear = newNode;
int dequeue() {
  if (queueFront == NULL) {
    printf("Queue is empty.\n");
    return -1;
  struct Node* temp = queueFront;
  int data = temp->data;
  queueFront = queueFront->next;
  if (queueFront == NULL) {
    queueRear = NULL;
  free(temp);
  return data;
void displayQueue() {
  struct Node* temp = queueFront;
  if (temp == NULL) {
    printf("Queue is empty.\n");
  } else {
    printf("Queue: ");
```

```
while (temp != NULL) {
       printf("%d", temp->data);
       temp = temp->next;
    printf("\n");
}
int main() {
  int choice, data;
  while (1) {
    printf("1. Push to Stack\n");
    printf("2. Pop from Stack\n");
    printf("3. Display Stack\n");
    printf("4. Enqueue to Queue\n");
    printf("5. Dequeue from Queue\n");
    printf("6. Display Queue\n");
    printf("7. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice) {
       case 1:
          printf("Enter data to push to stack: ");
          scanf("%d", &data);
          push(data);
          break:
       case 2:
         data = pop();
          if (data != -1) {
            printf("Popped from stack: %d\n", data);
          break:
       case 3:
          displayStack();
          break;
       case 4:
          printf("Enter data to enqueue to queue: ");
          scanf("%d", &data);
```

```
enqueue(data);
       break;
     case 5:
       data = dequeue();
       if (data != -1) {
          printf("Dequeued from queue: %d\n", data);
       break;
     case 6:
       displayQueue();
       break;
     case 7:
       exit(0);
     default:
       printf("Invalid choice\n");
  }
}
return 0;
```

```
Choose operation:

1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
7. Exit
Enter your choice: 1
Enter data to push: 1
1 is pushed to stack
Choose operation:
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
7. Exit
Enter data to push: 2
2 is pushed to stack
Choose operation:
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Stack
7. Exit
Enter your choice: 1
Enter data to push: 2
2 is pushed to stack
6. Display Queue
7. Exit
Enter your choice: 1
Enter data to push: 3
3 is pushed to stack
```

```
Choose operation:
1. Stack Push
 2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
 6. Display Queue
 7. Exit
Enter your choice: 3
Enter data to enqueue: 4
 Choose operation:
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
 7. Exit
Enter your choice: 3
Enter data to enqueue: 5
 Choose operation:
1. Stack Push
2. Stack Pop
 3. Queue Enqueue
4. Queue Dequeue
 5. Display Stack
 6. Display Queue
 7. Exit
 Enter your choice: 3
 Enter data to enqueue: 6
```

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

    Display Stack
    Display Queue

7. Exit
Enter your choice: 6
Queue: 4 -> 5 -> 6 -> NULL
Choose operation:

    Stack Push
    Stack Pop

3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
7. Exit
Enter your choice: 2
Popped element is 3
```

```
Choose operation:
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue

    Display Stack
    Display Queue

7. Exit
Enter your choice: 5
Stack: 2 -> 1 -> NULL
Choose operation:

    Stack Push
    Stack Pop

3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
7. Exit
Enter your choice: 4
Dequeued element is 4
Choose operation:
1. Stack Push
2. Stack Pop
3. Queue Enqueue
4. Queue Dequeue
5. Display Stack
6. Display Queue
7. Exit
Enter your choice: 6
Queue: 5 -> 6 -> NULL
```

WAP to Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list

```
(7) (NAP a DST Lab
 @ WAP to employent doubly link last with postoritive
     operations.
  @ Create a doubly Unked Est
  1 Insert a new nocle at the beginning
  @ Insert the node bared on a specific location.
 @ Insert the new node of the end
 @ Display the contents of the Cest.
us # Enclude astdlooh>
  # Enclude C Stdleboh>
  Struct Node &
           Ent data; a to to the state of the beautiful
           Struct Node & next ;
          Struct node * Prev;
  struct Node * Creade Node ( Ent data)
     struct node * new rock = (struct rode *) mode (struct rode);
      if (! newnode)
         Printf "Memory allocation Not pasible");
         seteion - NULL;
     new sdala = dala;
      new -x next = - NULLE,
     new -> prev = NOUL;
     return necomode;
```

```
Struct Noch to create lost ()
       h gettern NULL;
    void enserset At Bey (struct Nocle ** head, ent doda)
      struct node * neconode = creede node (data)
       il ( mhead = = NULL)
          shead = new node;
         ordarn:
      newnode > next = * head;
      ( *head ) -s prev = new Nocle;
      + head = neco nocle;
  votal Ensert ALOR (Struct Noale + + head, Put data, Put post
   I struct woode to neco node = creat woode (dota);
     [ [ * head == NULL 11 pos == 0)
          a gosent of Bog (nead, duta);
            return;
    Struct noch stemp = *nead;
         2
1);
     int count = 00
     while (tup!=NULL &x count < pos-1)
      d tup=tup snext;
          Count H;
    ? (temp = = NOLL)
       Possibly Possition is beyond the current list");
       Ensert At end (head, data);
      return;
   new Nock -> next - temp- next;
   if (tup shept != NULL)
       d tuop > next > prev = new mode;
      temp-s next = new node;
      new node - prev = temp;
3
```

```
vold ensent at end (struct wode >>> head, Ent dela)
      Strut node * new no de = Create no de (aceta);
       9[ [ * head = = NULL)
         & thead = new node;
           return;
       Struct Node teup = whead;
       while (temp-snext ! = NULL)
           d temp = tap > nept;
        tup -s next = new node;
        newnod sprev = temp;
vold display list (struct node * head)
   d of [ head == NULL)
       a proof[" Lat a enpty");
         orduan;
    struct wood & tunp = head;
    Print[ 1 hst contents: ");
    while ( temp! = NOCH)
        a printfly +d", & teup >data);
          temp= temp-snext;
 3
```

```
WAP to implement doubly linked list with
  primitive operations.
a) create a doubly linked list.
b) insert a new node at the beginning.
c) insert the node biased on a operation location
d) insert a new node at the end
e) display the content of the list
  Hindude & Stdio. hy
  # include Astdib.h>
   struct node &
    int data:
     struct node prev;
     struct node * next;
   struct node" creatilist() &
     return NULL;
   void insertat beg (struct node ** head, int data) {
     strut Node P newnode = (strut node* mallor (sike of lst
    rewnode → data = data
     Now node prevz NULL;
     newnode > next = * head;
    if ( *head = = NULL)
    { (* head) -> prev = newnode;}
    * hered = neunode;
```

```
Ent moetin ()
    Smut Node + head = createlest ();
     Entental Beg (shead, 10);
    Prosent 1 Beg ( & need , 20);
     Entrevit at end (should, 30);
     Putert At loc (8 nead 125, 2);
    desplay (head);
    returno;
3
  hast contents : 20
                       LO
                   ( brule memory allocation fallers )
                THE RESERVED - CONTRACTOR OF THE
                          Egota -tasa - stores
                              (got as won burt ?)
```

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data:
  struct Node* prev;
  struct Node* next:
};
struct Node* head = NULL;
void createList(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->prev = NULL;
  newNode->next = NULL;
  if (head == NULL) {
    head = newNode;
  } else {
    struct Node* temp = head;
    while (temp->next != NULL) {
      temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
  }
}
void insertLeft(int newData, int existingData) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = newData;
  struct Node* temp = head;
  while (temp != NULL && temp->data != existingData) {
    temp = temp->next;
  if (temp != NULL) {
    newNode->next = temp;
```

```
newNode->prev = temp->prev;
    if (temp->prev != NULL) {
       temp->prev->next = newNode;
     } else {
       head = newNode;
    temp->prev = newNode;
  } else {
    printf("Node with data %d not found.\n", existingData);
}
void deleteNode(int value) {
  struct Node* temp = head;
  while (temp != NULL && temp->data != value) {
    temp = temp->next;
  }
  if (temp != NULL) {
    if (temp->prev != NULL) {
       temp->prev->next = temp->next;
     } else {
       head = temp->next;
    if (temp->next != NULL) {
       temp->next->prev = temp->prev;
    free(temp);
    printf("Node with value %d deleted.\n", value);
  } else {
    printf("Node with value %d not found.\n", value);
}
void displayList() {
  struct Node* temp = head;
  if (temp == NULL) {
    printf("List is empty.\n");
```

```
} else {
     printf("Doubly Linked List: ");
     while (temp != NULL) {
       printf("%d ", temp->data);
       temp = temp->next;
    printf("\n");
}
int main() {
  int choice, data, existing Data;
  while (1) {
    printf("1. Create List\n");
    printf("2. Insert Node to the Left\n");
     printf("3. Delete Node\n");
    printf("4. Display List\n");
    printf("5. Exit\n");
     printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice) {
       case 1:
          printf("Enter data to create list: ");
          scanf("%d", &data);
          createList(data);
          break:
       case 2:
          printf("Enter new node data: ");
          scanf("%d", &data);
          printf("Enter the existing node data to insert left of: ");
          scanf("%d", &existingData);
          insertLeft(data, existingData);
          break;
       case 3:
          printf("Enter the node value to delete: ");
          scanf("%d", &data);
          deleteNode(data);
          break;
```

```
case 4:
    displayList();
    break;
    case 5:
        exit(0);
    default:
        printf("Invalid choice\n");
    }
}
return 0;
```

```
Enter the data:1
enter the data:1
enter 1 to continue:1
Enter the data:2
enter 1 to continue:1
Enter the data:3
enter 1 to continue:2
Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 1
Enter value to insert at the beginning: 4
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
 4. Display List
 5. Exit
Enter your choice: 4
4<->1<->2<->3<->NULL
nenu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
 5. Exit
Enter your choice: 2
Enter value to insert: 5
Enter the position:2
Nenu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
 5. Exit
Enter your choice: 4
4<->5<->1<->2<->3<->NULL
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 3
```

Menu:

- 1. Insert at Beginning
- 2. Insert at Specific Position
- 3. Insert at End
- 4. Display List
- 5. Exit

Enter your choice: 3

Enter value to insert at the end: 6

Menu:

- 1. Insert at Beginning
- 2. Insert at Specific Position
- 3. Insert at End
- 4. Display List
- 5. Exit

Enter your choice: 4

4<->5<->1<->2<->3<->6<->NULL

Write a program a) ToconstructabinarySearchtree. b) To traverse the tree using all the methods i.e., inorder, preorder and post order c) To display the elements in the tree

```
@ write a magrain
      @ To construct a broang search tree.
      To havere the tree wing all the methods
      @ To display the elevent in the tree.
    ws # Encluder stdon's
       # Encluder Stdlib-h>
       struct rocked
            Int duta:
            smut rode & left;
           struct made o payt;
      3,
     struct node * Erreate rode (in data)
       Struct nock & newrode = (struct nocks) malloc (size of (struct nock));
       newnode > dato = detai (on he have
        new rode - bft = NULL;
       new node - sight = NULL;
       return newrode;
  of (root == NULL)

L return creathrodd (doba);
      eluis [ set duto e root -> duta)

d root -> left - ment ( root + left , duta);

g eluis [ dato > root -> duta)

d root -> right = 8 nort ( root + right duta);

y
                                  5 had 1000 5 root = neck!
      return root
rold enordintraverot (shut node to root)
       int travel (root > left);
       print ("rd"; root s data);
       entr (root-) right);
      3 3 8 E
```

```
Noid Part Istruct node
    of Ilouti = NOTE)
        Printf(" 1.d", root > doda);
         Pat (root sleft);
        pot (root -s signe);
void Pot ( strul node to rat)
  ( if | sat! = NULL)
       ( pHP// soot states);
       Pot(noot > left);

Pot(noot - right);

Pot(noot - right);

Pot(noot - root - data);
    4
void diplay (struct node = root)
    if (200f; = NUFT)
                        eturn creiterchestatio);
      display root > loft):
        Pbl "7d", root sdated;
       displayfroot - righth, and the
 Ent math ()
    shud node + root = NULL;
     EASENET.
     moot = insert(noot, s)
     root = fosert (root, 3)
           = insert (root, 5)
     root = finent (root, 6)
      more intrivoot);
           Pat (200+):
           pot (moot)!
                                        2356
        retiano;
```

```
#include <stdio.h>
#include <stdlib.h>
// Structure for a node in the binary search tree
struct Node {
  int data;
  struct Node* left;
  struct Node* right;
};
// Function to create a new node
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->left = newNode->right = NULL;
  return newNode;
}
// Function to insert a node in the binary search tree
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;
}
// In-order traversal (Left, Root, Right)
void inorder(struct Node* root) {
  if (root != NULL) {
     inorder(root->left);
     printf("%d", root->data);
     inorder(root->right);
```

```
// Pre-order traversal (Root, Left, Right)
void preorder(struct Node* root) {
  if (root != NULL) {
     printf("%d", root->data);
     preorder(root->left);
     preorder(root->right);
}
// Post-order traversal (Left, Right, Root)
void postorder(struct Node* root) {
  if (root != NULL) {
     postorder(root->left);
     postorder(root->right);
     printf("%d ", root->data);
}
// Function to display the elements in the tree using in-order traversal
void display(struct Node* root) {
  printf("In-order traversal: ");
  inorder(root);
  printf("\n");
int main() {
  struct Node* root = NULL;
  int choice, data;
  while (1) {
     printf("1. Insert Node\n");
     printf("2. In-order Traversal\n");
     printf("3. Pre-order Traversal\n");
     printf("4. Post-order Traversal\n");
     printf("5. Display In-order Traversal\n");
     printf("6. Exit\n");
     printf("Enter your choice: ");
     scanf("%d", &choice);
     switch (choice) {
```

```
case 1:
        printf("Enter data to insert: ");
       scanf("%d", &data);
       root = insert(root, data);
        break;
     case 2:
       printf("In-order Traversal: ");
       inorder(root);
       printf("\n");
        break;
     case 3:
       printf("Pre-order Traversal: ");
       preorder(root);
       printf("\n");
       break;
     case 4:
       printf("Post-order Traversal: ");
       postorder(root);
        printf("\n");
       break;
     case 5:
       display(root);
       break;
     case 6:
       exit(0);
     default:
       printf("Invalid choice\n");
  }
}
return 0;
```

OUTPUT:

```
Binary Search Tree Operations:
1. Insert a node
2. Display In-order Traversal
3. Display Pre-order Traversal
4. Display Post-order Traversal
5. Exit
Enter your choice: 1
Enter the value to insert: 20
Enter your choice: 1
Enter the value to insert: 30
Enter your choice: 1
Enter the value to insert: 35
Enter your choice: 1
Enter the value to insert: 40
Enter your choice: 1
Enter the value to insert: 25
Enter your choice: 1
Enter the value to insert: 15
Enter your choice: 2
In-order Traversal: 15 20 25 30 35 40
Enter your choice: 3
Pre-order Traversal: 20 15 30 25 35 40
Enter your choice: 4
Post-order Traversal: 15 25 40 35 30 20
```

Program 9

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

OBSERVATION:

```
start, into)
 #Enclude 1 stdio. hs
Hindlele c con o. hs
# define max 100
Ent graph [MAN] [MAN] + VISI tw/[MAN);
out auce sman, front - 1, rear =
Void enquene cont node)
Las (rear = = MAX -1)
       randf ( Quem Full ");
     3 elu 0
       2 86 (poort == -1)
           front = 0;
        quem [rear] = node;
 ent dequeux ()
  & gut node;
    if ( front = = -1)
      * pour [ Queen Empty);
        return -1;
          no de = que (front);
          front ++;
         front = rear=-1;
  3
```

void bys (But start, Butn) enquencstast); Wenderstologo, hs VSPFED Pstert J= 1; while thant! = -1) I sot noch = dequest () , and I'd an a come to come again to point ((aga, node); for (fut 1:0; 82 n; 8++) Y if (graph (nod)] i7 == 1 & x 1 visited (i) enqueu (i); El (Econ Lecel " vasted tid =1; Ent main ()

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTICES 100
struct Queue {
  int items[MAX_VERTICES];
  int front, rear;
};
struct Graph {
  int adj[MAX_VERTICES][MAX_VERTICES];
  int vertices;
};
void initQueue(struct Queue* q) {
  q->front = -1;
  q->rear = -1;
int isQueueEmpty(struct Queue* q) {
  return q->front == -1;
}
void enqueue(struct Queue* q, int value) {
  if (q->rear == MAX_VERTICES - 1) {
    printf("Queue Overflow\n");
    return;
  if (q->front == -1)
    q->front = 0;
  q->rear++;
  q->items[q->rear] = value;
int dequeue(struct Queue* q) {
  if (isQueueEmpty(q)) {
    printf("Queue Underflow\n");
    return -1;
```

```
int item = q->items[q->front];
  q->front++;
  if (q->front > q->rear) {
    q->front = q->rear = -1;
  return item;
}
void initGraph(struct Graph* g, int vertices) {
  g->vertices = vertices;
  for (int i = 0; i < vertices; i++) {
    for (int j = 0; j < vertices; j++) {
       g->adj[i][j] = 0;
     }
  }
}
void addEdge(struct Graph* g, int u, int v) {
  g->adi[u][v] = 1;
  g->adj[v][u] = 1;
void bfs(struct Graph* g, int startVertex) {
  int visited[MAX_VERTICES] = \{0\};
  struct Queue q;
  initQueue(&q);
  visited[startVertex] = 1;
  enqueue(&q, startVertex);
  printf("BFS Traversal starting from vertex %d: ", startVertex);
  while (!isQueueEmpty(&q)) {
     int currentVertex = dequeue(&q);
    printf("%d", currentVertex);
    for (int i = 0; i < g->vertices; i++) {
       if (g->adj[currentVertex][i] == 1 &&!visited[i]) {
          enqueue(&q, i);
          visited[i] = 1;
```

```
printf("\n");
int main() {
  struct Graph g;
  int vertices, edges, u, v, startVertex;
  printf("Enter number of vertices: ");
  scanf("%d", &vertices);
  initGraph(&g, vertices);
  printf("Enter number of edges: ");
  scanf("%d", &edges);
  for (int i = 0; i < edges; i++) {
    printf("Enter edge (u v): ");
    scanf("%d %d", &u, &v);
    addEdge(&g, u, v);
  }
  printf("Enter the starting vertex for BFS: ");
  scanf("%d", &startVertex);
  bfs(&g, startVertex);
  return
```

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTICES 100
struct Graph {
  int adj[MAX_VERTICES][MAX_VERTICES];
  int vertices;
};
void initGraph(struct Graph* g, int vertices) {
  g->vertices = vertices;
  for (int i = 0; i < vertices; i++) {
     for (int j = 0; j < vertices; j++) {
       g->adj[i][j] = 0;
void addEdge(struct Graph* g, int u, int v) {
  g->adj[u][v] = 1;
  g->adj[v][u] = 1;
void dfs(struct Graph* g, int vertex, int visited[]) {
  visited[vertex] = 1;
  for (int i = 0; i < g->vertices; i++) {
     if (g->adj[vertex][i] == 1 \&\& !visited[i]) {
       dfs(g, i, visited);
     }
int isConnected(struct Graph* g) {
  int visited[MAX_VERTICES] = {0};
  dfs(g, 0, visited);
  for (int i = 0; i < g->vertices; i++) {
     if (!visited[i]) {
       return 0;
     }
```

```
return 1;
int main() {
  struct Graph g;
  int vertices, edges, u, v;
  printf("Enter number of vertices: ");
  scanf("%d", &vertices);
  initGraph(&g, vertices);
  printf("Enter number of edges: ");
  scanf("%d", &edges);
  for (int i = 0; i < edges; i++) {
     printf("Enter edge (u v): ");
     scanf("%d %d", &u, &v);
     addEdge(&g, u, v);
  if (isConnected(&g)) {
     printf("The graph is connected.\n");
  } else {
     printf("The graph is not connected.\n");
  }
  return 0;
```

OUTPUT:

```
Enter the number of vertices: 6
Enter the adjacency matrix:
0 1 1 0 1 1
1 0 0 1 0 1
0 1 1 1 0 0
1 1 1 0 1
1 1 1 0 0
Enter the starting vertex: 1
BFS Traversal: 1 0 3 5 2 4
```

```
Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 1 1
1 1 0
1 0 1 0
0 0 0 1
Enter the starting vertex: 0
DFS Traversal: 0 1 2 3
```