VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT on

Data Structures using C Lab

(23CS3PCDST)

Submitted by

Shaikh Uzair Ahmed (1BM23CS307)

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



This is to certify that the Lab work entitled "Data Structures using C Lab (23CS3PCDST)" carried out by **Shaikh Uzair Ahmed (1BM23CS307)**, 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. 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.

Lab faculty Incharge Namratha M
Assistant Professor
Department of CSE, BMSCE

Dr. Kavitha Sooda
Professor & HOD
Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	30/09/2024	Stack Implementation using arrays 4-	
2	07/10/2024	Infix to Postfix Conversion	8-12
3	15/10/2024	Queue implementation using arrays	13-16
4	21/10/2024	Circular Queue implementation using arrays	17-21
5	29/10/2024	Insertion Operation in Singly Linked List & Leet Code Valid Parenthesis	22-26
6	11/11/2024	Deletion Operation in Singly Linked List & Leet Code Daily Temperature	27-34
7	02/12/2024	(i) Multiple Operation in Singly Linked List (ii) Stack and Queue in Singly Linked List	35-44
8	09/12/2024	Insertion operation in Doubly Linked List	45-49
9			
10			

Github Link:

https://github.com/Shaikh-Uzair-Ahmed/1BM23CS307

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

	Stacks (Pseudo Code)		
	The state of the s		
-	- Initialize an arman a arm		
-	- make following functions - size = -1 - max = User defined four choice		
1	pust = max = User diffred John choice		
	98 Empty () 1 7 ()		
	- If \$72e == -1 return any [max-1]		
	return True		
	else drsplay()		
-	else display() return Falst for (9=@(Max-1), 920, 9)		
	90 = 111 (2		
-	98 F411 () print (are [7])		
	9f 897e == full		
	else True		
1150	yetum False		
	· Transfers + no		
166	Push (Hem)		
	Pf 25 F411 ()		
	schoon " Array Full"		
100	dse		
	1 Saze += 1		
	arr[size] = iten		
	(1) for at 2 2242 17 18000 /		
	Day ()		
	91 95 Empty ()		
	seturn "Array Empty"		
	eler return ar [Gaze-1]		
	195K7		
	68x-=1		

```
Code:
#include <stdio.h>
#include <stdlib.h>
int top = -1;
int isEmpty(int arr[]) {
  return top == -1;
int isFull(int arr[], int limit) {
  return top == limit - 1;
int Top(int arr[]) {
  if (isEmpty(arr)) {
     printf("Stack is empty.\n");
     return -1;
  }
  return arr[top];
void Display(int arr[]) {
  if (isEmpty(arr)) {
     printf("Stack is empty.\n");
     return;
  printf("Stack elements: ");
  for (int i = top; i >= 0; i--) {
     printf("%d", arr[i]);
  printf("\n");
void Push(int value, int arr[], int limit) {
  if (isFull(arr, limit)) {
     printf("Stack is full.\n");
  } else {
     top++;
     arr[top] = value;
     printf("Pushed %d onto the stack.\n", value);
  }
```

```
void Pop(int arr[]) {
  if (isEmpty(arr)) {
     printf("Stack is empty.\n");
  } else {
     printf("Popped %d from the stack.\n", arr[top]);
}
int main() {
  int limit;
  printf("Enter the limit of the stack: ");
  scanf("%d", &limit);
  int arr[limit];
  while (1) {
     int choice, value;
     printf("\nStack Operations:\n");
     printf("1. Push\n");
     printf("2. Pop \ ");
     printf("3. Top\n");
     printf("4. Display\n");
     printf("5. Exit\n");
     printf("Enter your choice: ");
     scanf("%d", &choice);
     switch (choice) {
       case 1:
          printf("Enter value to push: ");
          scanf("%d", &value);
          Push(value, arr, limit);
          break;
       case 2:
          Pop(arr);
          break;
       case 3:
          printf("Top element is: %d\n", Top(arr));
          break;
       case 4:
          Display(arr);
          break;
       case 5:
          exit(0);
       default:
```

```
printf("Invalid choice. Please try again.\n");
}
return 0;
```

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

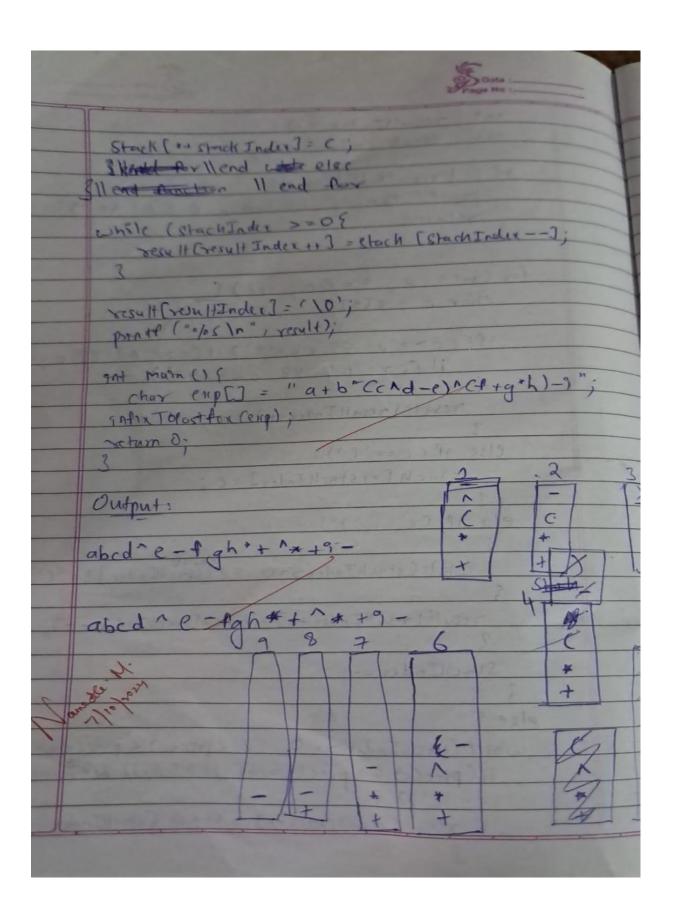
```
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 3
Stack is full.
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 4 from the stack.
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 4
Stack elements: 5
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 5
PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA>
```

Output 1 Output 2

Write a program to convert a given valid parenthesized in-fix arithmetic expression to post-fix expression. The expression consists of single character operands and the binary operators +,-,*,/.

```
90 th asthmetic expression to postful expression
 The expression consists of single character
Hinclyde < stdro-h>
# milude < std/26.4>
tranchede < strang. h >
       return -1;
char assocrativity (charc)
     IN TX To Post IX (const char #15) {
        lin = stelen(s)
  char result [en + 1];
  Char stack [len]
```

	Sonte:
	I ant resultInder = 0;
the stred	and Stack Index = -7;
2570n	AND PROPERTY AND ADDRESS OF THE PARTY OF THE
× ·	If (I result 11 I stack) {
7	prints ("Memory allocation field! In").
	- ? seturn;
1	1 (2 d 2 - 2 d 2 d 2 d 2 d 2 d 2 d 2 d 2 d
1	for (m+ 1=0; 9 < len; 9++) {
-	Char C = S[1];
1	1f((c>='a' 88 (<='z')11(c>='A' 88 c<='Z')
1	11 (C >= 'b') 88 C == (9'))
1	11 (c>='0' 88 (== (9'))
	result[resultIndex++] = c;
	3
	else of (== '(') {
	Stach[++stach Index] = C;
30	3 1
	else if (c == ')')
	whole (stach Index) = 0 22 stach Index] != ((')
	whole (stach Index) = 0 22 (stach Index) != ()
	1
	result [resultIndex ++] = Stack[StackIndex];
	3
	StachIndex;
	3
1 41	else {
	while (stach Index >= 0 22 (prec(c) < prec(stach (stach in))
	11 (prec(c) == prec(stach (stach I hour)) ec
	== (L')) {
ALC: UNKNOWN	== ([')) { sesult [recultIndex++] = Stack (chackIndex -];
-	3



```
Code:
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int prec(char c){
  if (c == '+' || c == '-') {
     return 1;
  if (c == '*' || c == '/') {
     return 2;
  if (c == '^')
     return 3;
  return 0;
char associativity(char c){
  if(c=='^')
     return 'R';
  return 'L';
}
void infixtopostfix(const char *s){
  int len = strlen(s);
  char result[len+1];
  char stack[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=s[i];
     if ((c)='a' && c<='z') || (c>='A' && c<='Z'))
       result[resultIndex++] = c;
     else if(c=='('){
       stack[++StackIndex]=c;
```

```
else if(c==')'){
       while (StackIndex>=0 && stack[StackIndex]!='(')
          result[resultIndex++]=stack[StackIndex--];
      StackIndex--;
     else{
       while (StackIndex>=0 && (prec(c))<prec(stack[StackIndex]) ||
(prec(c)==prec(stack[StackIndex]) && associativity(c)=='L'))
          result[resultIndex++]=stack[StackIndex--];
       stack[++StackIndex]=c;
  }
  while (StackIndex>=0)
     result[resultIndex++]=stack[StackIndex--];
  result[resultIndex++]='\setminus 0';
  printf("%s\n",result);
int main(){
  char \exp[] = "a+b*(c^d-e)^(f+g*h)-i";
  infixtopostfix(exp);
  return 0;
```

```
PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA> cd "c:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA\" ; if ($?) { gcc infi xpostfix.c -o infixpostfix } ; if ($?) { .\infixpostfix } abcd^e-fgh*+^*+i-
```

Output

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.

	Page No :
1	Program to Implement (INEAR QUEUE
1	# Trinclude < State . h >
1	#anclude < conso. h>
1	Hanclude <pre> <pre>process - h></pre></pre>
1	# define QUE_SIZE 5"
1	and often front = 0 rear = -1 g (QUE STREET
1	vord mount rear (Int Item, Int "rear , Int q [])
	1 4/1103
	of (scar == QUE_SIZE-1)
	Constant delicate
	prooff ("Queue Overflow In")
14.1	pront ("Queue Overflow In") return;
	7 1/20108 16 40 174 mass
	* 8tar = * rear + 1;
	Cfree 1 - Hem!
	of [rear] = Hem;
2	ant delete front (ant + front) ant * reas, ant q([])
3	ant delicerron
1-1	20000 1 2 2 2 2
14	of (front > * reax) return - 1;
* Come	ceturn of (front)++3;
	3
	vord display Q (sat front, ant rear, Int q (7)
6	S Manual
4	and i. College Marin
X	It (Boont Stear)
7	(
	106.5
	prantf (" queue 95 empty to ");
	return:
	3
	pronth (" contents of quear in")-
	for () = for all 1 acres and 1 m
	3 Printf("ofd Im", of [8]);
	3 ((d) (d) [6]);
	the state of the s

-	Page No:
	The second secon
-	Void magn ()
3	5
-	and charce;
-	clasere);
-	for (i;)
100	prentil (" In: 1: Insertrear In 2: delete Front In 3: displaying
-	Exist In ");
-	Scanf (" of od" Schotce)
-	Switch (choice)
-	Switch (Choice)
	Case 1: point P ("enter the stem to be inserted in")
-	Scanf ("of d", 2 aften);
-	Ensertreax (Hem, Exeax, q);
-	break;
	Case 2) Hem = delete front (2 front, 2 rear, q);
1000	of (refer == -1)
- 1112	
-	eise
-	
-	pronto ("Then deleted = glad) n" (term)
-	break;
	Case 3: displayo (tront rear, q1);
_	break;
	datale exit(0);
	& default: printil "Try again invalid input");
	331
	getch(); and almost the second of
	1
	Output, and a contract
	2: delete Ront 2. delete from
	3: Arsplay 3. drayles 4: ext
	h: ext
	I enter than to be treeted 5 4

```
Code:
#include <stdio.h>
#include <stdlib.h>
#define QUE_SIZE 5
int item, front = 0, rear = -1, q[N];
void insertrear(int item, int *rear, int q[]) {
  if (*rear == N - 1) {
     printf("Queue Overflow \n");
     return;
  *rear = *rear + 1;
  q[*rear] = item;
int Deque(int *front, int *rear, int q[]) {
  if (*front > *rear) {
     return -1;
  return q[(*front)++];
void displayQ(int front, int rear, int q[]) {
  if (front > rear) {
     printf("Queue is empty \n");
     return;
  printf("Contents of Queue:\n");
  for (int i = front; i \le rear; i++) {
     printf("%d \n", q[i]);
  }
}
int main() {
  int choice;
  for (;;) {
     printf("\n1: Insert rear\n2: Delete front\n3: Display\n4: Exit\n");
     scanf("%d", &choice);
```

```
switch (choice) {
     case 1:
       printf("Enter the item to be inserted: ");
       scanf("%d", &item);
       insertrear(item, &rear, q);
       break;
     case 2:
       item = Deque(&front, &rear, q);
       if (item == -1)
          printf("Queue is empty\n");
       else
          printf("Item deleted = %d \n", item);
       break;
     case 3:
       displayQ(front, rear, q);
       break;
     case 4:
       exit(0);
     default:
       printf("Invalid choice, please try again.\n");
  }
}
return 0;
```

```
1: Insert rear
2: Delete front
                                                               2: Delete front
3: Display
                                                               3: Display
4: Exit
                                                               4: Exit
Enter the item to be inserted: 10
                                                               Enter the item to be inserted: 2
1: Insert rear
2: Delete front
                                                               1: Insert rear
3: Display
                                                               2: Delete front
4: Exit
                                                               3: Display
                                                               4: Exit
Item deleted = 2
                                                               Enter the item to be inserted: 5
1: Insert rear
2: Delete front
                                                               1: Insert rear
3: Display
4: Exit
                                                               2: Delete front
                                                               3: Display
Item deleted = 5
                                                               4: Exit
1: Insert rear
                                                               Enter the item to be inserted: 2
2: Delete front
3: Display
                                                               1: Insert rear
4: Exit
                                                               2: Delete front
                                                               3: Display
Contents of Queue:
                                                               4: Exit
                                                              Enter the item to be inserted: 4
```

```
1: Insert rear
2: Delete front
3: Display
4: Exit
4
PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA>
```

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.

S Date :
LAD & Neula Queste
Handrale Estato. hs
module Estallib. h>
Allere N S
not hand or year or , or [N]
and threat
vord ensertrear (Ant Atem, Int rear, Pant que)
71 ((* rear +1) 0/0 N = = front) {
printe ("Quene over low")
return;
3
*rear = (*rear + 1) % N;
q[*reas] = 9tem;
Canada Inov
deque deletetron + (Int *front, got *rear , Int gel)
C CARAGE MAIN
A (+front == + sear == -1)
ad (Mant = -1 22 " rear = = -1)
prent (" Queue anderlan"); return -1;
prent (" queue anderlans"); return -]; else et (front = - Treax) {
* year = -1 -
return of ((Arent)
1 boat = 1
This temp = * front;
+ front = - 1
setura of Clemp); 3
Plac ((+ 1) ~ (+ 1)
retario and fortunated to N
plse Aront = ((+1sont+1)o/oN); return q [(+1sont+1)o/oN); return temp;
and the same of the same

vord alixplays (front mt front, and sear, and all 14 (trent == -1 28 scar == -1) Jehn front (" Que & Empty "); Print ("Vontents of June In");

for (9 = front; 94 = reax; 9 = (9+1) 0/0 N)

print ("%" (rears);

print ("%" (rears); void main () Southout = - 1; Entrais = -1; and charce; Clases () (for ();) printf (" In]: France In d: delletoon - In J. desplay In 41 ent In); Scanf ("ofed", Echarce) Switch (choice) case 1; MAH ("Enter 9ten to be Inserted"); Scanf ("ofed" Elter);

Instruct (often & rear of); Case 2: Them = ditable (2 front, 2 rear, 9) } of Oten 2=-1)

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

#define N 5

int q[N];

void Enqueue(int item, int *rear,int *front , int q[]) {
    if (((*rear+1)%N) == *front) {
        printf("Queue Overflow \n");
        return;
    }
    if (*front == -1)
    {
        *front = 0;
```

```
}
  *rear = ((*rear + 1)\% N);
  q[*rear] = item;
}
int Deque(int *front, int *rear, int q[]) {
  if (*front == -1 \&\& *rear == -1) {
     return -1;
  }
  int temp = q[*front];
  if(*front == *rear){
     *rear = -1;
     *front = -1;
  }
  else{
     *front = ((*front+1)\%N);
  }
  return temp;
}
void displayQ(int front, int rear, int q[]) {
  if (front == -1 \&\& rear == -1) {
     printf("Queue is empty \n");
     return;
  }
  printf("Contents of Queue:\n");
  for (int i = \text{front}; i != \text{rear}; i = (i+1)\% N) {
     printf("%d \n", q[i]);
  }
  printf("%d",q[rear]);
}
int main() {
  int choice;
  int rear = -1;
  int front = -1;
  int item;
  for (;;) {
     printf("\n1: Enque\n2: Deque1\n3: Display\n4: Exit\n");
     scanf("%d", &choice);
     switch (choice) {
```

```
case 1:
       printf("Enter the item to be inserted: ");
       scanf("%d", &item);
       Enqueue(item, &rear, &front, q);
       break;
     case 2:
       item = Deque(&front, &rear, q);
       if (item == -1)
          printf("Queue is empty\n");
       else
          printf("Item deleted = \% d \n", item);
       break;
     case 3:
       displayQ(front, rear, q);
       break;
     case 4:
       exit(0);
     default:
       printf("Invalid choice, please try again.\n");
return 0;
```

```
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 19
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 20
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 20
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 21
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 22
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 22
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter to Display
4: Exit
1: Display
4: Exit
1: Display
4: Exit
1: Display
4: Exit
```

```
1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 23

1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 24
Queue Overflow

1: Insert rear
2: Delete front
3: Display
4: Exit
1
Enter the item to be inserted: 24
Queue Overflow

1: Insert rear
2: Delete front
3: Display
4: Exit
3
Contents of Queue:
19
20
21
22
23
1: Insert rear
2: Delete front
3: Display
4: Exit
20
Item deleted = 19
```

```
1: Insert rear
2: Delete front
3: Display
4: Exit
Item deleted = 20
1: Insert rear
2: Delete front
3: Display
4: Exit
Contents of Queue:
23
1: Insert rear
2: Delete front
3: Display
4: Exit
PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA>
```

Write a program to Implement Singly Linked List with following operations

- a) Create a linked list.
- b) Insertion of a node at first position and at end of list.

Display the contents of the linked list.

Leetcode problem no.20 (Valid parantheses)

54
LAB-6 CUSTAG Double Page No.
panters)
#anclude <stdro.h></stdro.h>
#finclude Sstdlib h>
start Nude &
ant data;
struct Node " 120 N;
3;
struct Node " create Node (mt val)
al 1 at 10 Mal - Cal 1 At 1 a) W a
Struct Node * new Node = (struct Node ") malloc (street new Node -) data = val; (Struct Node)
ne Node -) In 1 = NULL; (Struct Node)
retyrn New Node;
3 Control and
Janton J. " De ENGLAND
vord display (struct Node + head) &
Struct Node ptx = head; while (ptx! = NULL) &
while (pto! = NULL) &
minth (ofed = ", IK -) datal;
by = bts -) leup!
S Challen S S
printh ("NULL In");
V90 151 61 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Void angestend (struct Node + " head and n) {
3+mct Node" femp = coeate Node (n); 2+ (head == NULL) {
head = temp;
Jelso ?
8 huch Node phr = head
While Cots = han 1 - MULL)
While (ptr =) han! = NULL) 3 pro-1/2 ptr = ptr -) lan 13

-	
	voted mont Front (shurt Mode" head, not n)
	Street Node + temp = create Node (n);
	head = temp;
	Wall should brook 3
	9nt main ()
	Shirl Node - head = NULL
	Shared Node - Nead - 1000
1 2	FN Chack val
1 10 10 21	unite (2) { low = a data a daman
	DYNH ("1. Insect Rooms (n");
	panta ("2-Insert at Rear In")
	" (" G. Extlan);
	" (" L. ENTHAN)
	Scant (" of d", 2 charce)
	switch (charce) ()
	Case 13 Case 13
	pf ("Enky value to greet at Front; ");
	Smit ("-lod", 2 val);
	morst Front (Thead, val);
	boeak)
	Case 22 May have been been
	meest a f year , ");
	Scant (g
	gnow End (Thead , val);
	loren 4.
	(25)
	drsplay (head)
	brea N;

	Case 6:	Page No :
	Case 4: exit(0): default: print (In who	chiece. please my again la 41)1
2	3	ing again in all
Re	ur O;	Assistant town

```
Code:
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* link;
};
struct Node* createNode(int val) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = val;
  newNode->link = NULL;
  return newNode;
}
void display(struct Node* head) {
  struct Node* ptr = head;
  while (ptr != NULL) {
     printf("%d -> ", ptr->data);
     ptr = ptr - link;
  }
  printf("NULL\n");
}
void insertEnd(struct Node** head, int n) {
  struct Node* temp = createNode(n);
  if (*head == NULL) {
     *head = temp; // If the list is empty, the new node becomes the head
  } else {
     struct Node* ptr = *head;
     while (ptr->link != NULL) {
       ptr = ptr->link;
     ptr->link = temp; // Add the new node at the end
  }
}
void insertFront(struct Node** head, int n) {
  struct Node* temp = createNode(n);
  temp->link = *head;
  *head = temp; // The new node becomes the head of the list
}
```

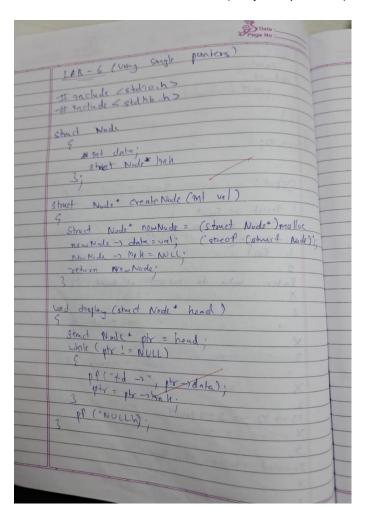
```
int main() {
  struct Node* head = NULL;
  int choice, val;
  while (1) {
     printf("1. Insert at Front\n");
     printf("2. Insert at Rear\n");
     printf("3. Display\n");
     printf("4. Exit\n");
     scanf("%d", &choice);
     switch (choice) {
        case 1:
          printf("Enter value to insert at front: ");
           scanf("%d", &val);
           insertFront(&head, val);
          break;
        case 2:
           printf("Enter value to insert at rear: ");
           scanf("%d", &val);
           insertEnd(&head, val);
          break;
        case 3:
           display(head);
          break;
        case 4:
           exit(0);
        default:
          printf("Invalid choice. Please try again.\n");
     }
  }
  return 0;
```

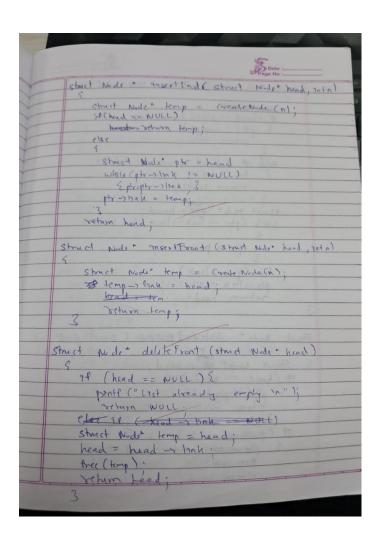
Leet Code Valid Parenthesis

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.

Leetcode Problem -- 739 (Daily Temperature)





```
start Node * delete End (street Node * hand)
       should better
       A (hand => NULL)
           pre ( smply );
       the of ( head - hah = = NULL)
        { tru (head);
        return NULL;
      Shart Node + 11 = head;
Struct Node* delete Art position ( struct Node* had,

Struct Node* = head;
  94 (head == NULL)
      return head:
```

- Br (m) 7=1; 12 posthon - 1 22 pto-> lank != NUL; 2 ptr - ptr - Iran; 98 (ptr-1991 == NULL) pf (" Out of bounds. In"). struct Nodo* temp = phs -> hah; phs-> lanh = temp-> lanh; tree (temp); 3. return & head; 994 mam () { struct Noder head : NULL; Int chace, val, pos: while (1) { pf ("1. Insert at Front In");

pf ("d. " " Rear m"); pt ("3. Delete Found In") Pf (" L. " Rear In "); pftus. " at Position In"); Pf ("6 - Nasylay In"); Scant ("olod", Zchorce); swatch (choice)

CORP (- Felox walve to trans) of front; head = Theres front (head, val); more at new ?" head = msextend (head, val); head = delefetrant (head); break; head = deleteted (head); Bredki; prints (" Enter position to delete: "); Scanf ("-/d", 2 pos); head = delete Atlasition (head, pos) breau; drsylay (hand); break; exist default:
7 Print ("In valid churce Try Again. In") 3 John o's

```
Code:
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* link;
};
// Function to create a new node
struct Node* createNode(int val) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = val;
  newNode->link = NULL;
  return newNode;
}
// Function to display the linked list
void display(struct Node* head) {
  struct Node* ptr = head;
  while (ptr != NULL) {
     printf("%d -> ", ptr->data);
     ptr = ptr->link;
  printf("NULL\n");
}
// Function to insert at the end of the list
struct Node* insertEnd(struct Node* head, int n) {
  struct Node* temp = createNode(n);
  if (head == NULL) {
     return temp; // If the list is empty, the new node becomes the head
  } else {
     struct Node* ptr = head;
     while (ptr->link != NULL) {
        ptr = ptr - link;
     ptr->link = temp; // Add the new node at the end
  }
  return head;
}
// Function to insert at the front of the list
struct Node* insertFront(struct Node* head, int n) {
  struct Node* temp = createNode(n);
  temp->link = head; // The new node points to the current head
                    // Return the new node as the new head of the list
  return temp;
```

```
}
// Function to delete the front node
struct Node* deleteFront(struct Node* head) {
  if (head == NULL) {
     printf("The list is already empty.\n");
     return NULL;
  }
  struct Node* temp = head;
  head = head->link; // Move head to the next node
  free(temp);
                    // Free the old head
  return head;
}
// Function to delete the end node
struct Node* deleteEnd(struct Node* head) {
  if (head == NULL) {
     printf("The list is already empty.\n");
     return NULL;
  if (head->link == NULL) { // Only one node in the list
     free(head);
     return NULL;
  struct Node* ptr = head;
  while (ptr->link->link != NULL) {
     ptr = ptr->link;
  free(ptr->link);
                    // Free the last node
  ptr->link = NULL; // Set the second last node's link to NULL
  return head;
}
// Function to delete a node at a specified position
struct Node* deleteAtPosition(struct Node* head, int position) {
  struct Node* ptr = head;
  if (head == NULL) {
     printf("The list is empty.\n");
     return NULL;
  if (position == 1) { // Delete the first node
     head=head->link;
     free(ptr);
     return head;
  }
  for (int i = 1; i < position - 1 && ptr->link != NULL; <math>i++) {
     ptr = ptr->link;
  if (ptr->link == NULL) {
```

```
printf("Position out of bounds.\n");
     return head;
  }
   struct Node* temp = ptr->link;
   ptr->link = temp->link;
   free(temp);
                     // Free the node at the specified position
   return head;
}
int main() {
   struct Node* head = NULL;
   int choice, val, pos;
   while (1) {
     printf("1. Insert at Front\n");
     printf("2. Insert at Rear\n");
     printf("3. Delete Front\n");
     printf("4. Delete Rear\n");
     printf("5. Delete at Position\n");
     printf("6. Display\n");
     printf("7. Exit\n");
     scanf("%d", &choice);
     switch (choice) {
        case 1:
           printf("Enter value to insert at front: ");
           scanf("%d", &val);
          head = insertFront(head, val);
          break;
        case 2:
           printf("Enter value to insert at rear: ");
           scanf("%d", &val);
          head = insertEnd(head, val);
          break;
        case 3:
          head = deleteFront(head);
          break;
        case 4:
          head = deleteEnd(head);
          break:
        case 5:
           printf("Enter position to delete: ");
           scanf("%d", &pos);
          head = deleteAtPosition(head, pos);
          break;
        case 6:
           display(head);
           break;
        case 7:
```

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

```
4 ∨int* dailyTemperatures(int* temperatures, int temperaturesSize, int* returnSize) {
        *returnSize = temperaturesSize;
        int* answer = (int*)calloc(temperaturesSize, sizeof(int));
        int* stack = (int*)malloc(temperaturesSize * sizeof(int));
        int top = -1;
10 V
        for (int i = 0; i < temperaturesSize; i++) {</pre>
11 V
            while (top !=-1 && temperatures[i] > temperatures[stack[top]]) {
12
                int j = stack[top--];
                answer[j] = i - j;
14
            stack[++top] = i;
16
        free(stack);
18
        return answer;
20 }
```

Leet Code Daily Temperatures

- Program 7

 a) Write a Program to Implement Single Link List with following operations:
 (i)Sort the linked list,
 (ii) Reverse the linked list,
 (iii)Concatenation of two linked lists.

a)	Stagle Isak 19st with Stack Stack Stack Reverse Concalenthan apartion Code: Attractic condicts attractic coldress.
a)	Stack - Sort - quene - Reverse - Concatenation approxion Code: Handred cetalio ho standade cetalio ho Stack That data;
a)	- Sort - Quene - Reverse - Concatenation approxion Code: Hinclude coldin to Handlude coldin ho Struct Node E The data;
a)	- Reverse - Concatenation approximation Code: Attractual certain ho attendade celebro ho Struct Node E and data;
a)	- Neverse - Concatenation approximation Code: Attractual certain ho attendade celabolish > Struct Node E and data;
a)	Code: Attractuck certain ho Atractude celdis ho Struct Node E The data;
a)	Code: Attractuck cortain to Atractude coldists.h > Struct Node Emy data;
	Street Node Em data;
	Struct Nocle E and data;
	and data;
	and data;
	ant data;
-	3; storet Wode* next;
* 3	3; when the table work about
	(and have
	Struct Nodo + Greate Nodo (not val)
	9
	Struct Node + newhode = (struct Node *) malloc (sneet
	newnode -> data = val; (etact Node))
	new node -> next = NUCC;
	seturn new-nodo;
	3 as well as you are made and all the
	void sort (street Note A hall)
	Void sort (Struct Node + head)
	Street Mode =7, 3;
	not temp;
	10x (9 = head - 9 - 2 mont 1
	Br (9 = head; 9 -> next 1 = NULL; 9 = 9 -> part)
	(for (y= T-> next; 1 = NOCC; 3- J-> next)
	71 (1-)data > 3-> data)

S Dote :
Street Node 1
S reverse Ma (6 had Node " head)
Street Node prev, rour, next n; prev = NULL;
carr = head;
nexin = head;
The state of the s
while (nexton 1 = NULL)
E JOHN STANDED THE MORE
next = next = next; con
curr-rouxt = prev;
been = com
3 Cury - next in;
P. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1. 2. 1.
3 hard press; setum press;
struct Node* (oncat (struct Node *head 1 struct Node *head 2) (S It (head 3 == NULL) return head 2;
m+ fount 20 (6 b sh) 32 25 45 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
Struct World & temp 2; " Tempo
temp 1 = head 2;
tem a hada;
while (temp1-) next! = NULL)
temp1 = temp2 -> Rest
temp1 -> next = head a;
AC III

Short Node " head;

Short Node " head;

If ("Noul 's");

Void > " ptr - date);

Void main ()

Short Node " head = NULL;

head > data = 4

head > next = Create Node (2);

head > next > next = create Node (9);

head > next > next = node (9);

head > next > next - next = (reate Node (1);

Short Node " head 2 = NULL;

head > data = 5 "

head > next > next = (reate Node (3);

head > next > next = (reate Node (3);

head > next > next = (reate Node (3);

head > next > next = (reate Node (3);

display (head);

chead = overse (head);

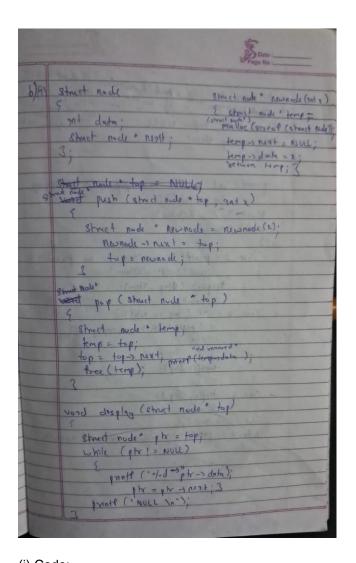
```
Code:
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* next;
};
struct Node* CreateNode(int val) {
  struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
  newnode->data = val;
  newnode->next = NULL;
  return newnode;
}
void sort(struct Node* head) {
  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;
          j->data = temp;
       }
     }
  }
}
struct Node* reverse(struct Node* head) {
  struct Node *prev = NULL, *curr = head, *nextn = NULL;
  while (curr != NULL) {
     nextn = curr->next;
     curr->next = prev;
     prev = curr;
     curr = nextn;
  }
  return prev;
}
struct Node* Concat(struct Node* head1, struct Node* head2) {
  if (head1 == NULL) return head2;
  struct Node* temp1 = head1;
  // Traverse to the last node of the first list
```

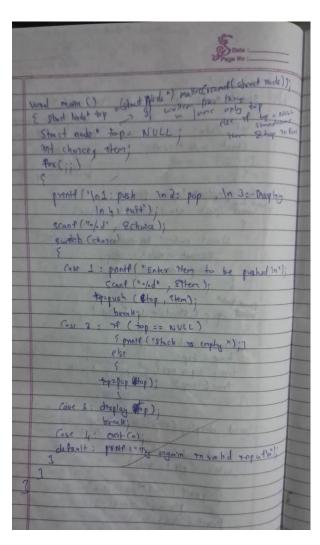
```
while (temp1->next != NULL) {
     temp1 = temp1->next;
  }
  temp1->next = head2;
  return head1:
}
void display(struct Node* head) {
  struct Node* ptr = head;
  while (ptr != NULL) {
     printf("%d ->", ptr->data);
     ptr = ptr->next;
  printf("NULL\n");
}
int main() {
  struct Node* head = CreateNode(4);
  head->next = CreateNode(2);
  head->next->next = CreateNode(9);
  head->next->next->next = CreateNode(1);
  struct Node* head2 = CreateNode(5);
  head2->next = CreateNode(3);
  head2->next->next = CreateNode(8);
  display(head);
  display(head2);
  sort(head);
  display(head);
  head2 = reverse(head2);
  display(head2);
  head = Concat(head, head2);
  display(head);
  return 0;
}
  PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307> cd "c:\Users\uzair\OneDrive\Des
 if ($?) { gcc SLLOperations.c -o SLLOperations } ; if ($?) { .\SLLOperations }
 4 ->2 ->9 ->1 ->NULL
 5 ->3 ->8 ->NULL
 1 ->2 ->4 ->9 ->NULL
 8 ->3 ->5 ->NULL
 1 ->2 ->4 ->9 ->8 ->3 ->5 ->NULL
 PS C:\Users\uzair\OneDrive\Desktop\1BM23CS307\DSA\Lab 8>
```

b)Write a Program to Implement Single Link List to simulate

(i)Stack

(ii)Queue Operations.





```
(i) Code:
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* CreateNode(int val) {
```

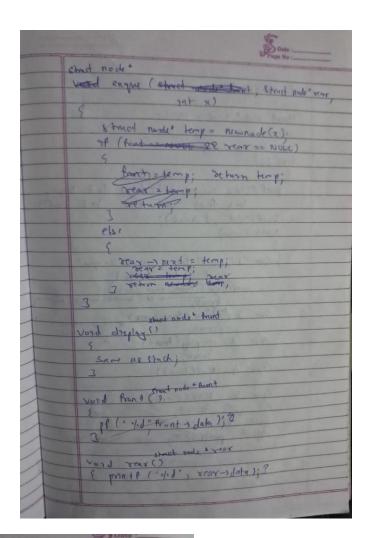
```
struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
  newnode->data = val;
  newnode->next = NULL;
  return newnode;
}
struct Node* push(struct Node*top,int x)
  struct Node* newnode = CreateNode(x);
  newnode->next = top;
  top = newnode;
  return newnode;
}
struct Node* pop(struct Node* top)
  struct Node* temp;
  temp = top;
  top = top->next;
  free(temp);
  return top;
}
void display(struct Node* head) {
  struct Node* ptr = head;
  while (ptr != NULL) {
     printf("%d ->", ptr->data);
     ptr = ptr->next;
  printf("NULL\n");
}
void main()
  struct Node* top = (struct Node*)malloc(sizeof(struct Node));
  top = NULL;
  int choice, item;
  for(;;)
     printf("\n1:push \n2:pop \n3:Display \n4:exit\n");
     scanf("%d",&choice);
     switch (choice)
     case 1:
        printf("Enter Item to be Pushed in Stack \n");
        scanf("%d",&item);
        top = push(top,item);
```

```
break;
     case 2:
        if (top == NULL)
          printf("Stack is Empty");
        else
        {
          top =pop(top);
        break;
     case 3:
        display(top);
        break;
     case 4:
        exit(0);
     default:
     printf("Try again invalid input \n");
     break;
     }
  }
}
```

```
| Strong | Company | Compa
```

Output (i)

1) Street node & gare as Stade Street nodo" new nucle (tool x) San as stack met nede " want deque (stourt node + front, street manting struct node + temps = front; At (front == NULL 87 mon NULL) print f ("Under low In"); return NULL; else of (Arang = reax) front = NIHT? fruit [/ old springed " tenp , daly); elec front = front - next; 3 return front



Page No :_ Vosd mas 1 () Street node front = NUCL) Street node + rear > NULL; Int chard, data; for (;;) prontf ("In 2: enque, m 2: deque in 3: Oraplay
In 4: front in S: Rear in 6: esof") Scanf ("ofod", 2 chosce); Switch (choice) Case 1: pront+ ("Enter Them to engine"); Scart Propody, 2data); rear enque (that, sees, data); NULL) [fine taken Case 2 that = word of (hout = word & Case 3: cheplay (tront); grear = NULL; Case 4: Pront (Pront); bean; Case S : Bear (sear); Case 6 : ent (o) 3 default: prate to Thy again mand inputility

```
(ii) Code:
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int data;
  struct Node* next;
};
struct Node* CreateNode(int val) {
  struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
  newnode->data = val;
  newnode->next = NULL;
  return newnode;
}
struct Node* enqueue(struct Node* rear, int x) {
  struct Node* newnode = CreateNode(x);
  if (rear == NULL) {
     return newnode;
  }
  rear->next = newnode;
  return newnode;
}
struct Node* dequeue(struct Node* front) {
  if (front == NULL) {
     printf("Queue is empty\n");
     return NULL;
  }
  struct Node* temp = front;
  front = front->next;
  free(temp);
  return front;
}
void display(struct Node* front) {
  struct Node* ptr = front;
  if (ptr == NULL) {
     printf("Queue is empty\n");
     return;
  while (ptr != NULL) {
     printf("%d ->", ptr->data);
     ptr = ptr->next;
  }
  printf("NULL\n");
}
```

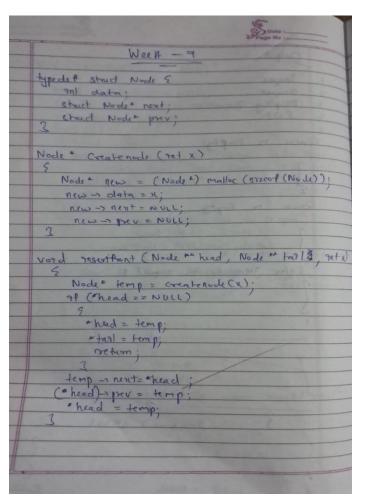
```
int main() {
  struct Node* front = NULL;
  struct Node* rear = NULL;
  int choice, item;
  for (;;) {
     printf("\n1: Enqueue\n2: Dequeue\n3: Display Queue\n4: Exit\n");
     scanf("%d", &choice);
     switch (choice) {
     case 1:
        printf("Enter item to be enqueued: ");
        scanf("%d", &item);
        rear = enqueue(rear, item); // Enqueue operation, rear updated
        if (front == NULL) {
          front = rear;
        break;
     case 2:
        front = dequeue(front); // Dequeue operation, front updated
        if (front == NULL) {
          rear = NULL;
        }
        break;
     case 3:
        display(front);
        break;
     case 4:
        exit(0); // Exit the program
     default:
        printf("Invalid input. Please try again.\n");
        break;
     }
  }
}
```

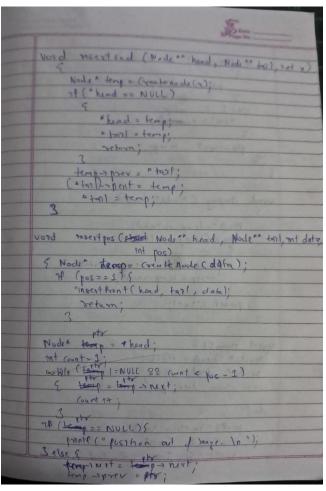
Output (ii)

Program 8

Write a Program 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





```
ph next ) = NULL)

sph next -> prev = temp;

sph -> ment = temp;

word drylog (Node head)

Node temp > head;

chill (temp / = NULL)

sphnith ("In")

yord tame) &

Node head = NULL;

Node tral = NULL;

node tral = NULL;

not chood, data, pue;

do &

prot ("In Menu in");

prot ("In Menu in ");

prot ("In Menu
```

```
Printer ( 3. Insent at Posshin in 1)
 prott (=4 - Drap by load In").
 prentl (" taken your (horas ");
Scant ("-jod", otherad;
 Switch (choice) [
  Case 1:
       prentil "Fater data to ment at front:
        scant ("o/od", 2data);
       gaser Front (8 head, 8tag), data);
       breaki
 Case 2:
        printf ("Fater data to ensest at end
         Scant ("ofad", Edata);
         Ingert End ( Shend, Ptoil, data);
         break;
 Cast 3:
       proof ("Enter position and data: ");
scanf ("Yel "fed", position and data: ");
toscof pos ( should, should, data)
         bred 1:
         prante (" List: ");
          desplay (hoad);
      profit ("exiting program. \n");
  depult:
  proff (" Invested Character");
in le (charce 1 = 5):
```

```
Code:
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
  int data:
  struct Node* prev;
  struct Node* next;
} Node;
Node* createNode(int data) {
  Node* temp = (Node*)malloc(sizeof(Node));
  temp->data = data;
  temp->prev = NULL;
  temp->next = NULL;
  return temp;
}
void insertFront(Node** head, Node** tail, int data) {
  Node* temp = createNode(data);
  if (*head == NULL) {
```

```
*head = *tail = temp;
  } else {
     temp->next = *head;
     (*head)->prev = temp;
     *head = temp;
  }
}
void insertEnd(Node** head, Node** tail, int data) {
  Node* temp = createNode(data);
  if (*tail == NULL) {
     *head = *tail = temp;
  } else {
     (*tail)->next = temp;
     temp->prev = *tail;
     *tail = temp;
  }
}
void insertAtPos(Node** head, Node** tail, int data, int pos) {
  Node* temp = createNode(data);
  if (pos == 1) {
     insertFront(head, tail, data);
     return;
  }
  Node* ptr = *head;
  int count = 1;
  while (ptr != NULL && count < pos - 1) {
     ptr = ptr->next;
     count++;
  }
  if (ptr == NULL) {
     printf("Position out of range.\n");
  } else {
     temp->next = ptr->next;
     temp->prev = ptr;
     if (ptr->next != NULL) {
        ptr->next->prev = temp;
     } else {
        *tail = temp;
     ptr->next = temp;
```

```
}
void printList(Node* head) {
  Node* ptr = head;
  while (ptr != NULL) {
     printf("%d ", ptr->data);
     ptr = ptr->next;
  }
  printf("\n");
}
int main() {
  Node* head = NULL;
  Node* tail = NULL;
  int choice, data, pos;
  do {
     printf("\nMenu:\n");
     printf("1. Insert at Front\n");
     printf("2. Insert at End\n");
     printf("3. Insert at Position\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 insert at front: ");
           scanf("%d", &data);
          insertFront(&head, &tail, data);
          break;
        case 2:
           printf("Enter data to insert at end: ");
           scanf("%d", &data);
          insertEnd(&head, &tail, data);
          break:
        case 3:
           printf("Enter position and data: ");
           scanf("%d %d", &pos, &data);
          insertAtPos(&head, &tail, data, pos);
          break:
        case 4:
           printf("List: ");
          printList(head);
           break;
```

```
printf("Invalid choice. Please try again.\n");
                      } while (choice != 5);
                       return 0;
                  }
                                                                 Enter your choice: 4
List: 3 12
                                                                                                                                    List: 3 12 5
1. Insert at Front
2. Insert at End
                                                                                                                                    1. Insert at Front
3. Insert at Position
                                                                 1. Insert at Front
                                                                                                                                    2. Insert at End
4. Display List
                                                                 2. Insert at End
                                                                                                                                    3. Insert at Position
5. Exit
                                                                 3. Insert at Position
                                                                                                                                    4. Display List
                                                                 4. Display List
Enter your choice: 1
                                                                                                                                    5. Exit
Enter data to insert at front: 12
                                                                 5. Exit
                                                                                                                                    Enter your choice: 3
                                                                 Enter your choice: 2
                                                                                                                                    Enter position and data: 2 4
                                                                 Enter data to insert at end: 5
1. Insert at Front
                                                                 Menu:
2. Insert at End
                                                                                                                                    1. Insert at Front
                                                                 1. Insert at Front
3. Insert at Position
                                                                                                                                    2. Insert at End
                                                                 2. Insert at End
4. Display List
                                                                                                                                    3. Insert at Position
                                                                 3. Insert at Position
5. Exit
                                                                 4. Display List
                                                                                                                                    4. Display List
Enter your choice: 1
                                                                                                                                    5. Exit
                                                                 5. Exit
Enter data to insert at front: 3
                                                                 Enter your choice: 4
List: 3 12 5
                                                                                                                                    Enter your choice: 4
                                                                                                                                    List: 3 4 12 5
                                                                 Menu:
1. Insert at Front
                                                                                                                                    Menu:
                                                                 1. Insert at Front
                                                                                                                                    1. Insert at Front
2. Insert at End
                                                                 2. Insert at End
                                                                                                                                    2. Insert at End
3. Insert at Position
                                                                 3. Insert at Position
                                                                                                                                    3. Insert at Position
4. Display List
                                                                 4. Display List
                                                                                                                                    4. Display List
5. Exit
                                                                 5. Exit
                                                                                                                                    5. Exit
Enter your choice: 4
                                                                 Enter your choice: 3
                                                                                                                                    Enter your choice: 5
List: 3 12
                                                                 Enter position and data: 2 4
                                                                                                                                    Exiting program.
```

case 5:

break; default:

printf("Exiting program.\n");

Output 1 Output 2 Output 3