Coure Name: Data Structures and Algorithms(CBS1003)

ASSESSMENT-1

24BBS0116

N.AKSHAY

1.

PSEUDOCODE:

```
START
Initialize stack[MAX] and top = -1
FUNCTION PUSH():
   IF top == MAX - 1:
      PRINT "Stack Overflow"
   ELSE:
      READ element
      INCREMENT top by 1
      stack[top] = element
FUNCTION POP():
   IF top == -1:
      PRINT "Stack Underflow"
   ELSE:
      DECREMENT top by 1
FUNCTION DISPLAY():
   IF top == -1:
      PRINT "Stack is empty"
   ELSE:
      FOR i = top TO 0:
         PRINT stack[i]
```

MAIN PROGRAM:

```
    WHILE TRUE:
        PRINT "1. PUSH  2. POP  3. DISPLAY  4. EXIT"
        READ choice
        SWITCH(choice):
            CASE 1: CALL PUSH()
            CASE 2: CALL POP()
            CASE 3: CALL DISPLAY()
            CASE 4: EXIT
            DEFAULT: PRINT "Invalid choice"
END
```

PROGRAM:

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

#define MAX 5

int stack[MAX], top = -1;

void push() {
    int element;
    if (top == MAX - 1) {
        printf("Stack Overflow!\n");
    } else {
        scanf("%d", &element);
        stack[++top] = element;
    }
}

void pop() {
    if (top == -1) {
```

```
        printf("Stack Underflow!\n");
    } else {
        top--;
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty.\n");
    } else {
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;
    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1: push(); break;
            case 2: pop(); break;
            case 3: display(); break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

OUTPUT:

```
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the element to push: 10
10 pushed onto the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the element to push: 20
20 pushed onto the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements are: 20 10
```

2.

PSEUDOCODE:

START

Initialize queue[MAX], front = -1, rear = -1

FUNCTION ENQUEUE():

IF rear == MAX - 1:

PRINT "Queue Overflow"

ELSE:

READ element

```
        IF front == -1:
            front = 0
        INCREMENT rear by 1
        queue[rear] = element
FUNCTION DEQUEUE():
    IF front == -1 OR front > rear:
        PRINT "Queue Underflow"
        front = -1, rear = -1
    ELSE:
        INCREMENT front by 1
FUNCTION DISPLAY():
    IF front == -1 OR front > rear:
        PRINT "Queue is empty"
    ELSE:
        FOR i = front TO rear:
            PRINT queue[i]
MAIN PROGRAM:
    WHILE TRUE:
        PRINT "1. ENQUEUE  2. DEQUEUE  3. DISPLAY  4. EXIT"
        READ choice
        SWITCH(choice):
            CASE 1: CALL ENQUEUE()
            CASE 2: CALL DEQUEUE()
            CASE 3: CALL DISPLAY()
            CASE 4: EXIT
END
PROGRAM:
#include <stdio.h>
#include <stdlib.h>
```

```
#define MAX 5

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

void enqueue() {
    int element;
    if (rear == MAX - 1) {
        printf("Queue Overflow\n");
    } else {
        scanf("%d", &element);
        if (front == -1) front = 0;
        queue[++rear] = element;
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow\n");
        front = -1;
        rear = -1;
    } else {
        front++;
    }
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty\n");
    } else {
        for (int i = front; i <= rear; i++) {
```

```
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;
    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1: enqueue(); break;
            case 2: dequeue(); break;
            case 3: display(); break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

OUTPUT:

Test cases 1,2,3:

```
Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 10
Enqueued 10.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 23
Enqueued 23.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Queue elements: 10 23

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Dequeued value: 10

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Queue elements: 23
```

```
Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 4
Exiting.
PS C:\Users\Karan\cprograms> gcc DSA2.c
PS C:\Users\Karan\cprograms> ./a.exe

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Queue Underflow! No elements to dequeue.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Queue is empty.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 4
Exiting.
PS C:\Users\Karan\cprograms> gcc DSA2.c
PS C:\Users\Karan\cprograms> ./a.exe

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 12
```

3.

PSEUDOCODE:

```
START
Initialize queue[MAX], front = -1, rear = -1
FUNCTION ENQUEUE():
   IF (front == 0 AND rear == MAX - 1) OR (rear + 1) % MAX == front:
      PRINT "Queue Overflow"
   ELSE:
      READ element
      IF front == -1:
         front = 0
      rear = (rear + 1) % MAX
      queue[rear] = element
FUNCTION DEQUEUE():
   IF front == -1:
      PRINT "Queue Underflow"
   ELSE:
      IF front == rear:
         front = -1, rear = -1
      ELSE:
         front = (front + 1) % MAX
FUNCTION DISPLAY():
   IF front == -1:
      PRINT "Queue is empty"
   ELSE:
      SET i = front
      WHILE i != rear:
         PRINT queue[i]
         i = (i + 1) % MAX
      PRINT queue[rear]
```

MAIN PROGRAM:

```
   WHILE TRUE:
      PRINT "1. ENQUEUE  2. DEQUEUE  3. DISPLAY  4. EXIT"
      READ choice
      SWITCH(choice):
         CASE 1: CALL ENQUEUE()
         CASE 2: CALL DEQUEUE()
         CASE 3: CALL DISPLAY()
         CASE 4: EXIT
END
```

PROGRAM:

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

#define MAX 5

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

void enqueue() {
   int element;
   if ((front == 0 && rear == MAX - 1) || (rear + 1) % MAX == front) {
      printf("Queue Overflow\n");
   } else {
      scanf("%d", &element);
      if (front == -1) front = 0;
      rear = (rear + 1) % MAX;
      queue[rear] = element;
   }
}
```

```
void dequeue() {
    if (front == -1) {
        printf("Queue Underflow\n");
    } else {
        if (front == rear) {
            front = -1;
            rear = -1;
        } else {
            front = (front + 1) % MAX;
        }
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty\n");
    } else {
        int i = front;
        while (i != rear) {
            printf("%d ", queue[i]);
            i = (i + 1) % MAX;
        }
        printf("%d\n", queue[rear]);
    }
}

int main() {
    int choice;
    while (1) {
        scanf("%d", &choice);
```

```
        switch (choice) {
            case 1: enqueue(); break;
            case 2: dequeue(); break;
            case 3: display(); break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

OUTPUT:

Test case:1

```
Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 35
Enqueued 35.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 67
Enqueued 67.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Circular Queue elements are: 35 67

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Dequeued value: 35

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Circular Queue elements are: 67
```

Test case 2:

```
Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 12
Enqueued 12.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 45
Enqueued 45.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 78
Enqueued 78.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 66
Enqueued 66.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
```

Test case 3:

```
Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Circular Queue Underflow! No elements to dequeue.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 55
Enqueued 55.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Dequeued value: 55

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Circular Queue Underflow! No elements to dequeue.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Circular Queue is empty.
```

4.

PSEUOCODE:

Head = NULL

11

Junction Insert B (val): Create new Node new Nod. date = val Head

new Node. next= Hand = New Node

Print " Value inserted in beginning

nal

Junction Insert E():

create new Node

new Node date- nalue new Node ment - NULL HEAD == NULL; Head New Node

else:

set temp = head

while temp. next != NULL;

temp = temp next!= NULL: temp-temp-next temp. next = ne Node

punt "value insuted at end"

Junction Inserst N (uals, pos.)

create new Node new Node date = value

is position == 1; nenode.next=Head head = new Nade

els set temp

Head

for <= 5 to position - 2:

temp = temp. next

if temp = NULL print "saved"

retum

newNode, next = temp. next

= new Node

temp next print (value insekted)

Junction delde B(): if HEAD NULL; print "empty"

set temp

Head

Head-Head nist

delde temp

print "Deleted value from beginning "77

delite

Junction () {

if head == NULL

punt "List is empty"

Hij if head. next = NULL delete HEAD VALUE

Junction (search trednal); sit = Had = deship nature, while temp. data = valu

14 Buick &

}

temp!= NULL

return temp = turp. next

print ("Naluefind a 2019

temp = lemp. nest

pient'

null;

"List is empty

display (): 1 head == sao

else:

et is head != number

is time temp = NULL;

white teamp != NULL

pline temp date temp = tempy need

main program

while.

Disply and

punt menu (Insert, Delete, search exis input choice

switch choud

case 1:

input ualu (CAA) stippe style

Insut B (ual)

лана:

Insert E (ual)

лецез

nalue, position

case 4:

Delete B (G

case 5:

pdetiE (14

classes

case 6:

Input position

Delete position input positions

delete_position (position

cass 7:

case 8

сана

Input mal search (val)

display()

print" exiting

Buak"

pa exiting cus Innald croce

default:

Print >> "I nvalid choice"

Program:

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

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

struct Node *head = NULL;

void insertAtBeginning(int value) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = head;
    head = new_node;
}

void insertAtEnd(int value) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = NULL;
```

```
    if (head == NULL) {
        head = new_node;
    } else {
        struct Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new_node;
    }
}

void insertAtPosition(int value, int position) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    if (position == 1) {
        new_node->next = head;
        head = new_node;
        return;
    }
    struct Node *temp = head;
    for (int i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
```

```
    }
    if (temp == NULL) {
        printf("Position out of bounds\n");
    } else {
        new_node->next = temp->next;
        temp->next = new_node;
    }
}

void deleteAtBeginning() {
    if (head == NULL) {
        printf("List is empty\n");
    } else {
        struct Node *temp = head;
        head = head->next;
        free(temp);
    }
}

void deleteAtEnd() {
    if (head == NULL) {
        printf("List is empty\n");
```

```
    } else if (head->next == NULL) {
        free(head);
        head = NULL;
    } else {
        struct Node *temp = head;
        while (temp->next->next != NULL) {
            temp = temp->next;
        }
        free(temp->next);
        temp->next = NULL;
    }
}

void deleteAtPosition(int position) {
    if (head == NULL) {
        printf("List is empty\n");
    } else if (position == 1) {
        struct Node *temp = head;
        head = head->next;
        free(temp);
    } else {
        struct Node *temp = head, *prev = NULL;
```

```
        for (int i = 1; i < position && temp != NULL; i++) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Position out of bounds\n");
        } else {
            prev->next = temp->next;
            free(temp);
        }
    }
}

void search(int value) {
    struct Node *temp = head;
    int position = 1;
    while (temp != NULL) {
        if (temp->data == value) {
            printf("Element found at position %d\n", position);
            return;
        }
        temp = temp->next;
```

```
        position++;
    }
    printf("Element not found\n");
}

void display() {
    if (head == NULL) {
        printf("List is empty\n");
    } else {
        struct Node *temp = head;
        while (temp != NULL) {
            printf("%d -> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
    int choice, value, position;
    while (1) {
        scanf("%d", &choice);
```

```
switch (choice) {
    case 1:
        scanf("%d", &value);
        insertAtBeginning(value);
        break;
    case 2:
        scanf("%d", &value);
        insertAtEnd(value);
        break;
    case 3:
        scanf("%d %d", &value, &position);
        insertAtPosition(value, position);
        break;
    case 4:
        deleteAtBeginning();
        break;
    case 5:
        deleteAtEnd();
        break;
    case 6:
        scanf("%d", &position);
        deleteAtPosition(position);
```

```
            break;
        case 7:
            scanf("%d", &value);
            search(value);
            break;
        case 8:
            display();
            break;
        case 9:
            exit(0);
      }
   }
   return 0;
}
```

Output:

Testcase-1

```
Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 10
Enter value to insert at beginning: Inserted 10 at the beginning

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 2 20
Enter value to insert at end: Inserted 20 at the end.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 3 30 2
Enter value and position to insert: Inserted 30 at position 2.
```

```
Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 4
Deleted 10 from the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 8
List elements are: 30 20

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 9
Exiting...
```

Testcase-2:

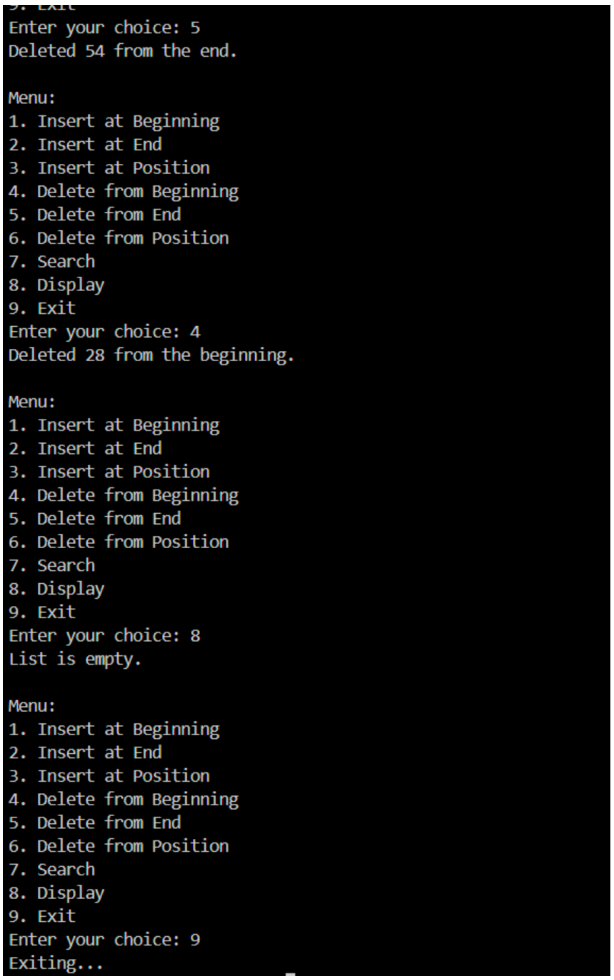
```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 23
Enter value to insert at beginning: Inserted 23 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 33
Enter value to insert at beginning: Inserted 33 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 7 88
Enter value to search: 88 not found in the list.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
```

```
Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 7 56
Enter value to search: 56 not found in the list.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 8
List elements are: 33 23

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 9
Exiting...
PS C:\Users\Karan\cprograms> gcc DSA4.c
PS C:\Users\Karan\cprograms> ./a.exe
```

Testcase-3:

```
Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 28
Enter value to insert at beginning: Inserted 28 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 2 54
Enter value to insert at end: Inserted 54 at the end.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 3 45 2
Enter value and position to insert: Inserted 45 at position 2.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
```

```
7. Search
8. Display
9. Exit
Enter your choice: 8
List elements are: 28 45 54

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 6 2
Enter position to delete: Deleted 45 from position 2.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 8
List elements are: 28 54

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 5
```



5.

Pseudocode:

Initialize stack as an array

Node { data

}

prev next

Head = null

Junction insert B(mal): create new Node new Nede data-mala newNodi prev = Head is head != NULL:

head. prev = newNode

Head = newNode

print "value inserted at the beginning"

Junction insert E (ual):

create new Node

newNod data = nal

new Node. next = NULL

if Head == NULL: newNode.pw = NULL

else:

Head - NiwNode

set temp = head while temp. next != NULL; temp = timp next

temp. next = new Node new Node. prev = temp

print "Value inserted at the end"

Junction insert N (val, pos);

create new Node newNode.data = value if pos==1:

new, Node next = Head

new wode. prev = NULL if Head != NULL ! head prev = new Node Head = newNode

else:

ret temp = Head for (int i=1; i <= pos-2; i++){

4(temp == NULL):

punt "Inualid" timp = timp next

3 return if temp == NULL:

print "Inualid" retun

newNode next=temp next. new Node prev = temp if temp. next bonNULL: temp. next.prev = new Node

temp. ment=

= new Node

quint "Value inserted at position"

Junction delele B():

if Head?

else:

== NULL !

punt "List is empty"

set temp = Head Head = Head next

if Head != NULL ! Head prev = NULL

delete temp

punt "Delite malu"

function delete E():

if Head

== NULL:

print "List is Empty" else if Head next == NULL Delete head

head = NULL

punt "Deleted malue from the end"

else:

set temp = Head

while temp. next != NULL; temp = temp next

temp. pr. next = NULL

Delete temp

print "Deleted value at end"

Junction delete N (pos); if head == NULL: print "List is Empty else if position == 1: set timp plead Head = Head. next

else:

Head != NULL:

Head. pew

C

NULL

Delete long

print "Deleted value from position"

sit temp = Head

for i = 1; i <= pos- 1; i++; if temp == NULL: punt "Invalid"

retun

temp = temp. next

if temp == NULL: funt "Snualid"

```
retuun
if temp next != NULL:
temp next.prev = temp prev
is temp. pur!= NULL:
temp. prev.next = limp. next
delete temp
print "Deleted from position.
Junction search (val):
set temp = Head
set pos =
= 1
while temp != NULL:
print "value found at position", pos
if temp data
==
Value!
retun
temp
leng time next
++ pos
print "Value not found"
function display ()!
if Head == NULL;
print "hist is empty"
elu:
set temp = Head
print "elements are:"
while temp != NULL ! print temy dala temp = timp next
Main program
while:
```

print menu:

1. Invit at Be start

2. Insut at end

3. Insut at pos

4. Delete at start

5. Delete at end

6. Delete at pos

7. search

8. Display

9. Exit

Input (choice)

case 1: insut B(); case 2:

case 3:

insut E();

insut N();

лан 4:

Mott B();

case 5;

delete();

лань:

case 7:

delete N(pos);

search (value);

лан 8:

display()

case 9:

retun

Default

"punt "Invalid"

Program:

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

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

struct Node *head = NULL;

void insertAtBeginning(int value) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->prev = NULL;
    new_node->next = head;
    if (head != NULL) {
        head->prev = new_node;
    }
    head = new_node;
}
```

```
void insertAtEnd(int value) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = NULL;
    if (head == NULL) {
        new_node->prev = NULL;
        head = new_node;
    } else {
        struct Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = new_node;
        new_node->prev = temp;
    }
}

void insertAtPosition(int value, int position) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    if (position == 1) {
```

```
    new_node->prev = NULL;
    new_node->next = head;
    if (head != NULL) {
        head->prev = new_node;
    }
    head = new_node;
    return;
}
struct Node *temp = head;
for (int i = 1; i < position - 1 && temp != NULL; i++) {
    temp = temp->next;
}
if (temp == NULL) {
    printf("Position out of bounds\n");
    free(new_node);
} else {
    new_node->next = temp->next;
    if (temp->next != NULL) {
        temp->next->prev = new_node;
    }
    temp->next = new_node;
    new_node->prev = temp;
```

```
    }
}

void deleteAtBeginning() {
    if (head == NULL) {
        printf("List is empty\n");
    } else {
        struct Node *temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        free(temp);
    }
}

void deleteAtEnd() {
    if (head == NULL) {
        printf("List is empty\n");
    } else if (head->next == NULL) {
        free(head);
        head = NULL;
```

```
    } else {
        struct Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->prev->next = NULL;
        free(temp);
    }
}

void deleteAtPosition(int position) {
    if (head == NULL) {
        printf("List is empty\n");
    } else if (position == 1) {
        struct Node *temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        free(temp);
    } else {
        struct Node *temp = head;
```

```
        for (int i = 1; i < position && temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Position out of bounds\n");
        } else {
            if (temp->next != NULL) {
                temp->next->prev = temp->prev;
            }
            if (temp->prev != NULL) {
                temp->prev->next = temp->next;
            }
            free(temp);
        }
    }
}

void search(int value) {
    struct Node *temp = head;
    int position = 1;
    while (temp != NULL) {
        if (temp->data == value) {
```

```
            printf("Element found at position %d\n", position);
            return;
        }
        temp = temp->next;
        position++;
    }
    printf("Element not found\n");
}

void display() {
    if (head == NULL) {
        printf("List is empty\n");
    } else {
        struct Node *temp = head;
        while (temp != NULL) {
            printf("%d <-> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}
```

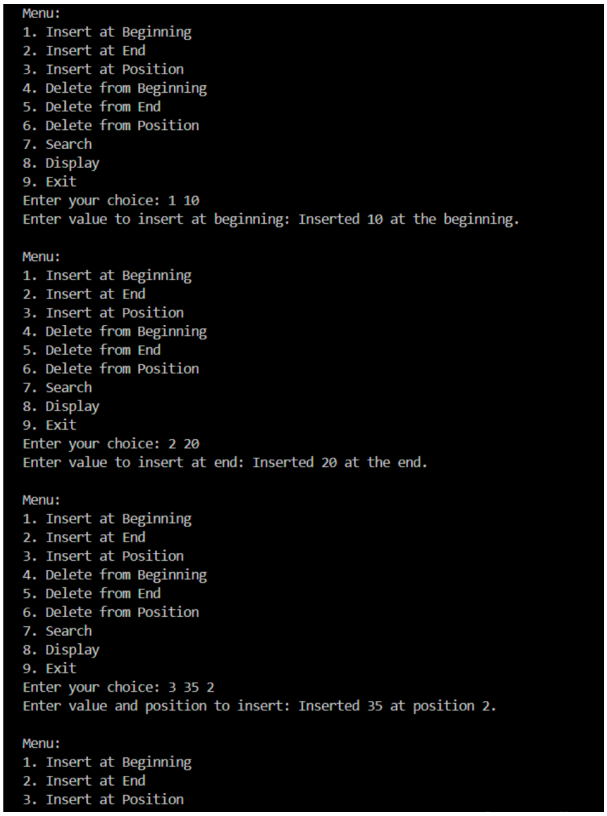
```
int main() {
    int choice, value, position;
    while (1) {
        printf("\n1. Insert at Beginning\n2. Insert at End\n3. Insert at
Position\n");
        printf("4. Delete at Beginning\n5. Delete at End\n6. Delete at
Position\n");
        printf("7. Search\n8. Display\n9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insertAtBeginning(value);
                break;
            case 2:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insertAtEnd(value);
                break;
            case 3:
                printf("Enter value and position: ");
```

```
            scanf("%d %d", &value, &position);
            insertAtPosition(value, position);
            break;
        case 4:
            deleteAtBeginning();
            break;
        case 5:
            deleteAtEnd();
            break;
        case 6:
            printf("Enter position to delete: ");
            scanf("%d", &position);
            deleteAtPosition(position);
            break;
        case 7:
            printf("Enter value to search: ");
            scanf("%d", &value);
            search(value);
            break;
        case 8:
            display();
            break;
```

```
            case 9:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Output:

Testcase-1:



```
Enter your choice: 8
List elements are: 10 35 20

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 4
Deleted 10 from the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 8
List elements are: 35 20

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 9
Exiting...
```

Testcase-2:

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 23
Enter value to insert at beginning: Inserted 23 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 45
Enter value to insert at beginning: Inserted 45 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 7 44
Enter value to search: 44 not found in the list.
```

Testcase-3:

```
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 2 50
Enter value to insert at end: Inserted 50 at the end.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 1 60
Enter value to insert at beginning: Inserted 60 at the beginning.

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete from Position
7. Search
8. Display
9. Exit
Enter your choice: 3 70 2
Enter value and position to insert: Inserted 70 at position 2.
```