DSA Digital Assignment

Name: Karan Nahta

Roll Num: 24BBS0085

Qn 1

Testcase – 1:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5

int stack[MAX];
int top = -1;
void push(int value);
int pop();
```

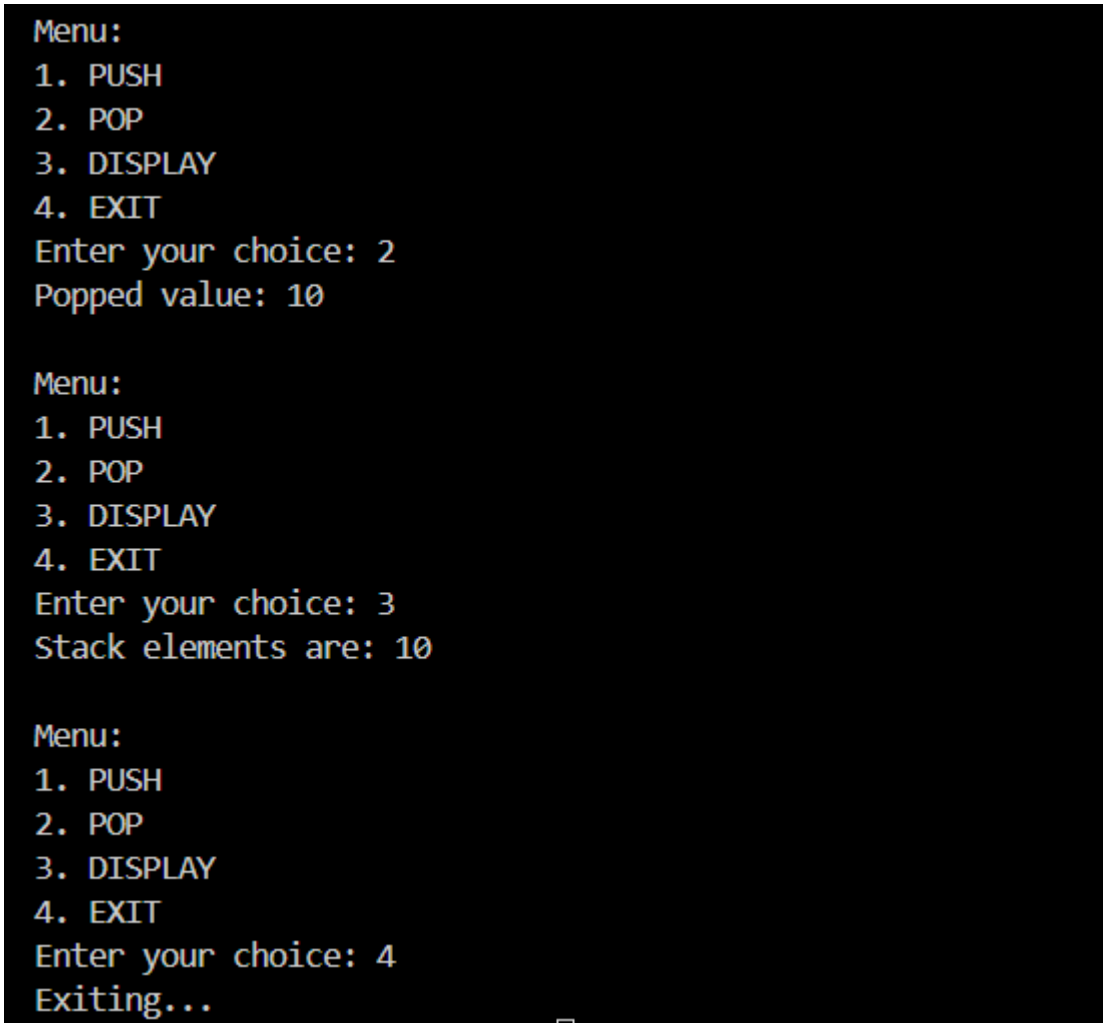
```
void display();

int main() {
    int choice, value;
    do {
        printf("\nMenu:\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 value to
push: ");
                scanf("%d", &value);
```

```
            push(value);
            break;
        case 2:
            value = pop();
            if (value != -1)
                printf("Popped value: %d\n",
value);
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Please
try again.\n");
    }
  }
```

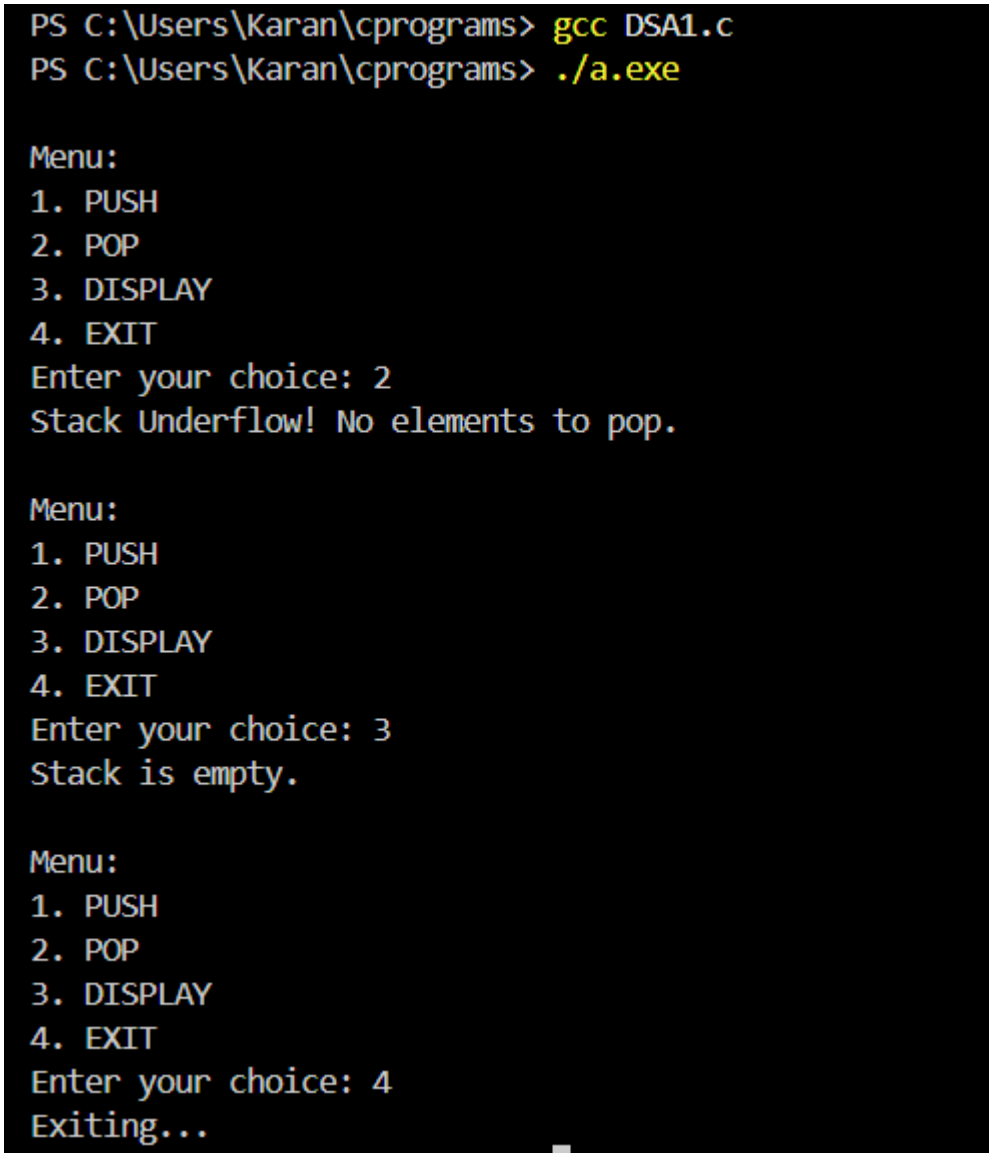
```
    while (choice != 4);
    return 0;
}

void push(int value) {
    if (top == MAX - 1) {
        printf("Stack Overflow! Cannot
push %d.\n", value);
    } else {
        stack[++top] = value;
        printf("Pushed %d onto the
stack.\n", value);
    }
}

int pop() {
    if (top == -1) {
        printf("Stack Underflow! No
elements to pop.\n");
```

```
        return -1;
    } else {
        return stack[top--];
    }
}

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

```
PS C:\Users\Karan\cprograms> gcc DSA1.c
PS C:\Users\Karan\cprograms> ./a.exe

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 10
Pushed 10 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 10
Pushed 10 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
Stack elements are: 10 10

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 2
Popped value: 10

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
```


Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 2
Popped value: 10
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
Stack elements are: 10
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 4
Exiting...

Testcase – 2:


PS C:\Users\Karan\cprograms> gcc DSA1.c
PS C:\Users\Karan\cprograms> ./a.exe
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 2
Stack Underflow! No elements to pop.
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
Stack is empty.
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 4
Exiting...

Testcase – 3:

```
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 12
Pushed 12 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 23
Pushed 23 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 134
Invalid choice! Please try again.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 24
Pushed 24 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
```

```
Enter the value to push: 45
Pushed 45 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 56
Pushed 56 onto the stack.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 67
Stack Overflow! Cannot push 67.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
Stack elements are: 56 45 24 23 12

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to push: 33
Stack Overflow! Cannot push 33.

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
```

```
Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 3
Stack elements are: 56 45 24 23 12

Menu:
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter your choice: 4
Exiting...
PS C:\Users\Karan\cprograms>
```

Question 2:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int queue[MAX];
int front = -1, rear = -1;
void enqueue(int value);
int dequeue();
void display();
```

```
int main() {
    int choice, value;
    do {
        printf("\nMenu:\n");
        printf("1. ENQUEUE\n");
        printf("2. DEQUEUE\n");
        printf("3. DISPLAY\n");
        printf("4. EXIT\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to
enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
```

```
            value = dequeue();
            if (value != -1)
                printf("Dequeued value:
%d\n", value);
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting.\n");
            break;
        default:
            printf("Invalid choice! Please
try again.\n");
        }
    } while (choice != 4);
    return 0;
}
```

```
void enqueue(int value) {
    if (rear == MAX - 1) {
        printf("Queue Overflow! Cannot
enqueue %d.\n", value);
    } else {
        if (front == -1)
            front = 0;
        queue[++rear] = value;
        printf("Enqueued %d.\n", value);
    }
}

int dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow! No
elements to dequeue.\n");
        return -1;
    } else {
        return queue[front++];
```

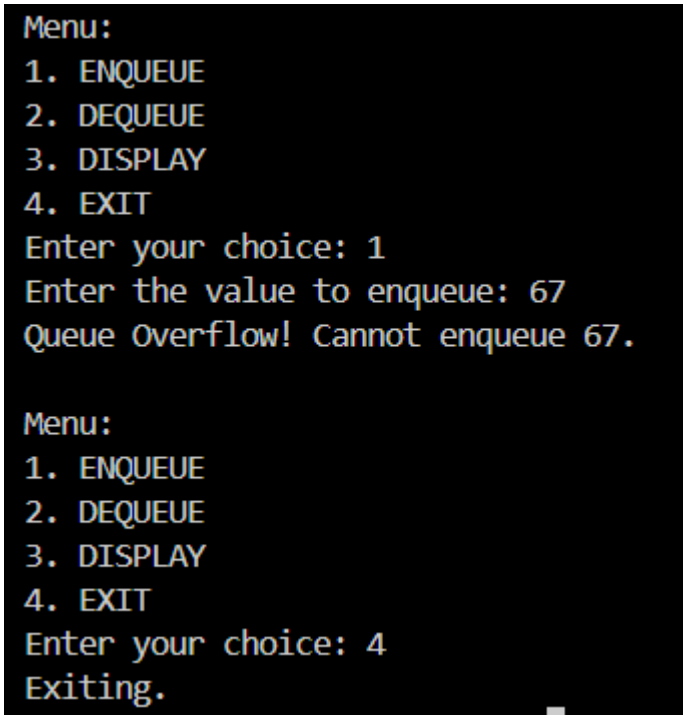
```
    }
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}
```

Testcase 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
```

```
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: 1
Enter the value to enqueue: 34
Enqueued 34.

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: 55
Enqueued 55.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 67
Invalid choice! Please try again.
```



Question 3

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int circularQueue[MAX];
int front = -1, rear = -1;
void enqueue(int value);
int dequeue();
void display();
```

```
int main() {
    int choice, value;

    do {
        printf("\nMenu:\n");
        printf("1. ENQUEUE\n");
        printf("2. DEQUEUE\n");
        printf("3. DISPLAY\n");
        printf("4. EXIT\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to
enqueue: ");
                scanf("%d", &value);
```

```
            enqueue(value);
            break;
        case 2:
            value = dequeue();
            if (value != -1)
                printf("Dequeued value:
%d\n", value);
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Please
try again.\n");
    }
  } while (choice != 4);
```

```
    return 0;
}

void enqueue(int value) {
    if ((front == 0 && rear == MAX - 1) ||
(rear + 1 == front)) {
        printf("Circular Queue Overflow!
Cannot enqueue %d.\n", value);
    } else {
        if (front == -1) // First element to
enqueue
            front = 0;
        rear = (rear + 1) % MAX;
        circularQueue[rear] = value;
        printf("Enqueued %d.\n", value);
    }
}
```

```
int dequeue() {
    if (front == -1) {
        printf("Circular Queue Underflow!
No elements to dequeue.\n");
        return -1;
    } else {
        int value = circularQueue[front];
        if (front == rear) { // Queue
becomes empty after this dequeue
            front = -1;
            rear = -1;
        } else {
            front = (front + 1) % MAX;
        }
        return value;
    }
}

void display() {
```

```
    if (front == -1) {
        printf("Circular Queue is
empty.\n");
    } else {
        printf("Circular Queue elements
are: ");
        int i = front;
        while (1) {
            printf("%d ", circularQueue[i]);
            if (i == rear)
                break;
            i = (i + 1) % MAX;
        }
        printf("\n");
    }
}
```

Testcase – 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
```

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

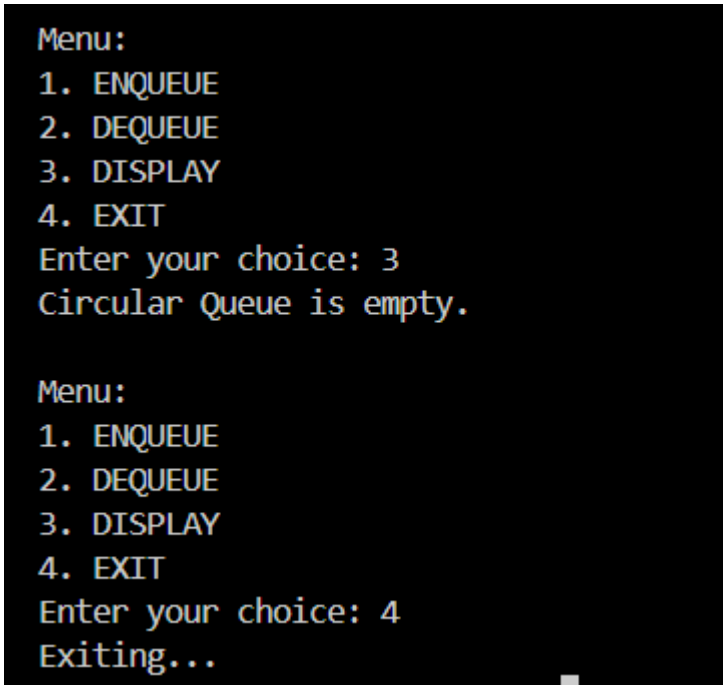
Testcase – 2:

```
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
```

```
Enter the value to enqueue: 89
Enqueued 89.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Dequeued value: 12

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 2
Dequeued value: 45

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 1
Enter the value to enqueue: 69
Enqueued 69.

Menu:
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
Enter your choice: 3
Circular Queue elements are: 78 66 89 69

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

Testcase – 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.
```



Question 4

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* head = NULL;
```

```
void insertB(int value);
void insertE(int value);
void insertN(int value, int pos);
void deleteB();
void deleteE();
void deleteN(int pos);
void search(int value);
void display();
int main() {
    int choice, value, pos;
    do {
        printf("\nMenu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from
Beginning\n");
        printf("5. Delete from End\n");
```

```
        printf("6. Delete from Position\n");
        printf("7. Search\n");
        printf("8. Display\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to insert at
beginning: ");
                scanf("%d", &value);
                insertB(value);
                break;
            case 2:
                printf("Enter value to insert at
end: ");
                scanf("%d", &value);
                insertE(value);
                break;
```

```
            case 3:
                printf("Enter value and
position to insert: ");
                scanf("%d%d", &value, &pos);
                insertN(value, pos);
                break;
            case 4:
                deleteB();
                break;
            case 5:
                deleteE();
                break;
            case 6:
                printf("Enter position to
delete: ");
                scanf("%d", &pos);
                deleteN(pos);
                break;
            case 7:
```

```
            printf("Enter value to search:
");
            scanf("%d", &value);
            search(value);
            break;
        case 8:
            display();
            break;
        case 9:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Try
again.\n");
        }
    }
    while (choice != 9);
    return 0;
}
```

```
void insertB(int value) {
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
    printf("Inserted %d at the
beginning.\n", value);
}

void insertE(int value) {
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
```

```
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("Inserted %d at the end.\n",
value);
}

void insertN(int value, int pos) {
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;

    if (pos == 1) {
        newNode->next = head;
        head = newNode;
```

```
    } else {
        Node* temp = head;
        for (int i = 1; i < pos - 1 && temp !=
NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Invalid position!\n");
            free(newNode);
            return;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    printf("Inserted %d at position
%d.\n", value, pos);
}

void deleteB() {
```

```
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    Node* temp = head;
    head = head->next;
    printf("Deleted %d from the
beginning.\n", temp->data);
    free(temp);
}

void deleteE() {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    if (head->next == NULL) {
        printf("Deleted %d from the
end.\n", head->data);
```

```
        free(head);
        head = NULL;
        return;
    }
    Node* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    printf("Deleted %d from the end.\n",
temp->next->data);
    free(temp->next);
    temp->next = NULL;
}

void deleteN(int pos) {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
```

```
    }
    if (pos == 1) {
        Node* temp = head;
        head = head->next;
        printf("Deleted %d from position
%d.\n", temp->data, pos);
        free(temp);
        return;
    }
    Node* temp = head;
    for (int i = 1; i < pos - 1 && temp !=
NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next ==
NULL) {
        printf("Invalid position!\n");
        return;
    }
```

```
    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    printf("Deleted %d from position
%d.\n", toDelete->data, pos);
    free(toDelete);
}

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

```
    printf("%d not found in the list.\n",
value);
}

void display() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    Node* temp = head;
    printf("List elements are: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
```

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
```

```
9. Exit
Enter your choice: 8
List elements are: 10 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: 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
```

```
9. Exit
Enter your choice: 5
Deleted 54 from 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: 4
Deleted 28 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 is empty.

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>
```

Question 5

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;
Node* head = NULL;
void insertB(int value);
```

```
void insertE(int value);
void insertN(int value, int pos);
void deleteB();
void deleteE();
void deleteN(int pos);
void search(int value);
void display();

int main() {
    int choice, value, pos;

    do {
        printf("\nMenu:\n");
        printf("1. Insert at
Beginning\n");
        printf("2. Insert at End\n");
```

```
        printf("3. Insert at
Position\n");
        printf("4. Delete from
Beginning\n");
        printf("5. Delete from
End\n");
        printf("6. Delete from
Position\n");
        printf("7. Search\n");
        printf("8. Display\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
            printf("Enter value to
insert at beginning: ");
            scanf("%d", &value);
            insertB(value);
            break;
        case 2:
            printf("Enter value to
insert at end: ");
            scanf("%d", &value);
            insertE(value);
            break;
        case 3:
            printf("Enter value and
position to insert: ");
            scanf("%d%d", &value,
&pos);
            insertN(value, pos);
```

```
            break;
        case 4:
            deleteB();
            break;
        case 5:
            deleteE();
            break;
        case 6:
            printf("Enter position to
delete: ");
            scanf("%d", &pos);
            deleteN(pos);
            break;
        case 7:
            printf("Enter value to
search: ");
```

```
            scanf("%d", &value);
            search(value);
            break;
        case 8:
            display();
            break;
        case 9:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice!
Try again.\n");
        }
    } while (choice != 9);
    return 0;
}
```

```
void insertB(int value) {
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = head;
    if (head != NULL) {
        head->prev = newNode;
    }
    head = newNode;
    printf("Inserted %d at the
beginning.\n", value);
}

void insertE(int value) {
```

```
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

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

```
    printf("Inserted %d at the
end.\n", value);
}

void insertN(int value, int pos) {
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->data = value;

    if (pos == 1) {
        newNode->next = head;
        newNode->prev = NULL;
        if (head != NULL) {
            head->prev = newNode;
        }
        head = newNode;
```

```
    } else {
        Node* temp = head;
        for (int i = 1; i < pos - 1 &&
temp != NULL; i++) {
            temp = temp->next;
        }
        if (temp == NULL) {
            printf("Invalid
position!\n");
            free(newNode);
            return;
        }
        newNode->next = temp-
>next;
        newNode->prev = temp;
        if (temp->next != NULL) {
```

```
            temp->next->prev =
newNode;
        }
        temp->next = newNode;
    }
    printf("Inserted %d at position
%d.\n", value, pos);
}

void deleteB() {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    Node* temp = head;
    head = head->next;
```

```
    if (head != NULL) {
        head->prev = NULL;
    }
    printf("Deleted %d from the
beginning.\n", temp->data);
    free(temp);
}

void deleteE() {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    if (head->next == NULL) {
        printf("Deleted %d from the
end.\n", head->data);
```

```
        free(head);
        head = NULL;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->prev->next = NULL;
    printf("Deleted %d from the
end.\n", temp->data);
    free(temp);
}

void deleteN(int pos) {
    if (head == NULL) {
```

```
        printf("List is empty!\n");
        return;
    }
    if (pos == 1) {
        Node* temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        printf("Deleted %d from
position %d.\n", temp->data,
pos);
        free(temp);
        return;
    }
    Node* temp = head;
```

```
    for (int i = 1; i < pos && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Invalid position!\n");
        return;
    }
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }
```

```
    printf("Deleted %d from
position %d.\n", temp->data,
pos);
    free(temp);
}

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

```
    }
    printf("%d not found in the
list.\n", value);
}

void display() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    Node* temp = head;
    printf("List elements are: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
```

```
    printf("\n");
}
```

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 35 2
Enter value and position to insert: Inserted 35 at position 2.

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

```
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.

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

```
Enter your choice: 5
Deleted 23 from 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: 6 1
Enter position to delete: Deleted 45 from position

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 is empty.

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 – 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.

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

```
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: 60 70 50

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
Enter position to delete: 8
Invalid position!

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>
```

Question ①

```
Initialize stack as an array of size max
set top = -1
function push(val):
    if top == max-1:
        print "Stack Overflow"
    else
    ++top
      stack[top] = val
      print "Value pushed"

function pop():
    if top == -1:
       print "Stack Underflow"
       return -1
    else:
       value = stack[top]
       --top
       return value

function display():
     if top == -1:
        print "Stack is empty"
     else
        for (i=Top; i>=0; i--):
           print stack[i]

Main program:
    while:
          print Menu options (1:PUSH, 2:POP, 3:DISPLAY, 4:EXIT)
          Input choice
          Switch choice
            case 1:
               Input value
               push(value)
            case 2:
               pop()
            case 3:
               display()
            case 4:
               print "exiting"
            Default:
               print "Invalid"
```

Question ②

```
Initialize queue as an array
set front = -1, rear = -1

function enqueue(val):
    if rear = max - 1:
        print "Queue Overflow"
    else
        if front == -1:
            front = 0

        ++rear
        queue(rear) = val
        print "Value Enqueued"

function dequeue():
    if front == -1 || front > rear:
        print "Queue underflowed"
        return -1
    else:
        value = queue[front]
        ++front
        return value

function display():
    if front == -1 || front > rear
        print "Empty"
    else:
        for(i = front; i < rear; i++):
            print queue[i]

main program:
    while:
        print menu option (1: Enqueue, 2: Dequeue, 3: Display, 4: Exit)
        input choice
        switch choice:
            case 1:
                input value
                enqueue(val)
            case 2:
                dequeue()
            case 3:
                display()
            case 4:
                print "Existing"
            Default:
                print "Invalid"
```

Question ③

```
Initialize circqueue as array
set front = -1, rear = -1
function enqueue (val ):
        if (front == 0 && rear = max - 1)||(rear + 1 == front ):
           print "queue overflow "
        else :
           if front = = -1 :
              front = 0
           rear = (rear + 1 ) %. max
           circqueue [rear ] = value
           print "Value enqueued "
function dequeue ( ):
        if front = = -1 :
           print "queue underflow "
           return -1
        ~~return~~ < 1
        else:
           val = circqueue [front ]
           if front == rear :
              front = -1 , rear = -1
           else :
              front = (front + 1 ) %. max
           return value
function display( ):
      if front = = -1 :
         print "empty circqueue"
      else :
         set i = front
         while :
            print circ-queue [i ]
               if i = = max :
                  break ;
               i = (i+1 ) %. max
main program :
   while :
      print Menu (1: Enqueue, 2: Dequeu
                  3: Display, 4: Exit ):
      input choice :
      switch choice :
        case 1 :
           input val
           enqueue (val )
        case 2 :
           Dequeue ( )
        case 3 :
           display ( )
        case 4 :
           print "Existing "
        Default :
           print "Invalid"
```

Question ④

```
Head = NULL

function insertB(val):
    Create newNode
    newNode.data = val
    newNode.next = Head
    Head = NewNode
    Print " Value inserted in beginning

function insertE(val):
    create newNode
    newNode.data = value
    newNode.next = NULL
    if HEAD == NULL;
        Head = NewNode
    else:
        set temp = head
        while temp.next != NULL:
            temp = temp.next != NULL:
                temp = temp.next
                temp.next = newNode
                print "Value inserted at end"

function insertN (val, pos):
    create newNode
    newNode.data = value
    if position == 1;
        newnode.next = Head
        head = newNode
    else set temp = Head
        for i = 1 to position - 2:
            temp = temp.next
            if temp = NULL
                print "Invalid"
                return
        newNode.next = temp.next
        temp.next = newNode
        print (Value inserted)

function deleteB():
    if HEAD = NULL;
        print "Empty"
    else
        set temp = Head
        Head = Head.next
        delete temp
        print " Deleted value from beginning"

function deleteB() {
    if head == NULL
        print "List is empty"
    if head.next = NULL
        delete HEAD
        VALUE

function (search tried val):
    set temp = Head =
    while temp.data = value
    while {
        temp != NULL
    }
    return temp = temp.next
    print ("Value found 2019
    temp = temp.next

display ( ): if head == null;
    print " List is empty

else:
    if head != number
    if temp = NULL;
    while temp != NULL
    print temp.data
    temp = temp.next
```

```
main program
    while:                                  Disply and
        print menu (Insert, Delete, Search, exit
        input choice
        switch choice
            case 1:
                input value (CAA) stupid style
                insertB(val)
            case 2:
                insertE(val)
            case3:
                value, position
            case 4:
                DeleteB();
            case 5:
                DeleteE();
            case 6:
                input position

Delete position
    input positions

delete_position (positions
    case 7:
        input val
        search(val)
    case 8
        display()
    case 9
        print "exiting Break"
    pa exiting cus Invalid choice
    default:
        Print >> "Invalid choice"
```

Question ⑤

```
Initialize stack as an array
Node {
    data
    prev
    next
}
Head = null

function insertB(val):
    create newNode
    newNode.data = value
    newNode.prev = Head
    if head != NULL:
        head.prev = newNode
    Head = newNode
    print "Value inserted at the beginning"

function insertE(val):
    create newNode
    newNod.data = val
    newNode.next = NULL
    if Head == NULL:
        newNode.prev = NULL
        Head = NewNode
    else:
        set temp = head
        while temp.next != NULL:
            temp = temp.next
        temp.next = newNode
        newNode.prev = temp
    print "Value inserted at the end"

function insertN(val, pos):
    create newNode
    newNode.data = value
    if pos == 1:
        newNode.next = HEad
        newNode.prev = NULL
        if Head != NULL:
            head.prev = newNode
        Head = newNode
    else:
        set temp = Head
        for (int i = 1; i <= pos-2; i++) {
            if (temp == NULL):
                print "Invalid"
            temp = temp.next
        } return
        if temp == NULL:
            print "Invalid"
            return
        newNode.next = temp.next
        newNode.prev = temp
        if temp.next != NULL:
            temp.next.prev = newNode
        temp.next = newNode
        print "Value inserted at position"

function deleteB():
    if Head == NULL:
        print "List is empty"
    else:
        set temp = Head
        Head = Head.next
        if Head != NULL:
            Head.prev = NULL
        delete temp
        print "Delete value"

function deleteE():
    if Head == NULL:
        print "List is Empty"
    else if: Head.next == NULL:
        Delete head
        head = NULL
        print "Deleted value from the end"
    else:
        set temp = Head
        while temp.next != NULL:
            temp = temp.next
        temp.prev.next = NULL
        Delete temp
        print "Deleted value at end"
```

```
function deleteN(pos):
    if head == NULL:
        print "List is Empty"
    else if position == 1:
        set temp = Head
        Head = Head.next
        if Head != NULL:
            Head.prev = NULL
        Delete temp
        print "Deleted value from position"

    else:
        set temp = Head
        for i = 1; i <= pos - 1; i++;
            if temp == NULL:
                print "Invalid"
                return
        temp = temp.next
        if temp == NULL:
            print "Invalid"
            return
        if temp.next != NULL:
            temp.next.prev = temp.prev
        if temp.prev != NULL:
            temp.prev.next = temp.next
        delete temp
        print "Deleted from position"

function search(val):
    set temp = Head
    set pos = 1
    while temp != NULL:
        if temp.data == Value:
            print "Value found at position", pos
            return
        temp = temp.next
        ++pos
    print "Value not found"
```

```
function display():
    if Head == NULL:
        print "List is empty"
    else:
        set temp = Head
        print "Elements are:"
        while temp != NULL:
            print temp.data
            temp = temp.next
```

```
Main program
  while:
    print menu:
    1. Insert at Beginning
    2. Insert at End
    3. Insert at pos
    4. Delete at start
    5. Delete at end
    6. Delete at pos
    7. search
    8. Display
    9. Exit
    Input(choice)
      case 1:
        insertB();
      case 2:
        insertE();
      case 3:
        insertN();
      case 4:
        deleteB();
      case 5:
        deleteE();
      case 6:
        deleteN(pos);
      case 7:
        search(value);
      case 8:
        display()
      case 9:
        return
      Default
        print "Invalid"
```