COMPUTER LAB

NAME: **SOUMYADIP MAITY**

ROLL NO.: **22053029**

SEC: **CSE-49**

YEAR: **2023-24**

ASSIGNMENT – 8

1. Write a menu driven program to implement Deques (both Input-restricted and Output-restricted) operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using array.

Input Restricted:

```
#include <stdio.h>
#include <stdbool.h>
int queue[20], i, data, R=-1, F=-1, size;
void enqueue(int data, int size){
if (R==size-1){
printf("\n Queue overflow\n");
}
if (F==-1&&R==-1){
R=F=0;
queue[R]=data;
}
else{
queue[++R]=data;
}
}
void dequeueF(){
if (F==-1&&R==-1){
printf("\n underflow \n");
}
else{
queue[F]=0 ;
F=F+1;
if(F>R){
F=-1;
R=-1;
}
}
}
void dequeueR(){
if (F==-1&&R==-1){
printf("\n underflow \n");
}
else{
queue[R]=0;
R-=1;
if(F>R){
F=-1;
R=-1;
}
}
}void peek(){
printf("\n %d \n", queue[F]);
}
void isEmpty(){
if (F==-1&&R==-1){
printf("The queue is empty \n");
```

```
}
else{
printf("The queue is not empty \n");
}
}
void isFull(){if (R==size-1){
printf("\n The queue is full \n");
}
else{
printf("\n The queue is not full \n");
}
}
void display(){
for (i=0; i<=size-1; i++){
printf("\n %d \n", queue[i]);
}
}
int main(){
int choice;
printf("\n Enter the size of the queue: ");
scanf("%d", &size);
while(true){
printf("\n 1. enqueue\n");
printf("2. dequeue from front\n");
printf("7. dequeue from rear\n");
printf("3. peek\n");
printf("4. display\n");
printf("5. check if queue is empty\n");
printf("6. check if queue is full\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice){
case 1:
printf("\n Enter the element: ");
scanf("%d", &data);
enqueue(data, size);
break;
case 2:
dequeueF();
break;
case 3:
peek();
break;
case 4:display();
break;
case 5:
isEmpty();
break;
case 6:
isFull();
break;
case 7:
dequeueR();
break;
default:
printf("Invalid choice\n");
}
}
return 0;
}
```

Output restricted:

```
#include <stdio.h>
#include <stdbool.h>
int queue[20], i, data, R=-1, F=-1, size;
void enqueueR(int data, int size){
if (R==size-1){
printf("\n Queue overflow\n");
}
if (F==-1&&R==-1){R=F=0;
queue[R]=data;
}
else{
queue[++R]=data;
}
}
void enqueueF(int data, int size){
if (F==0){
printf("\n Queue overflow\n");
}
if (F==-1&&R==-1){
R=F=0;
queue[F]=data;
}
else{
queue[--F]=data;
}
}
void dequeue(){
if (F==-1&&R==-1){
printf("\n underflow \n");
}
queue[F]=0;
F=F+1;if(F>R){
F=-1;
R=-1;
}
}
void peek(){
printf("\n %d \n", queue[F]);
}
void isEmpty(){
if (F==-1&&R==-1){
```

```
printf("The queue is empty \n");
}
else{
printf("The queue is not empty \n");
}
}
void isFull(){
if (R==size-1){
printf("\n The queue is full \n");
}
else{
printf("\n The queue is not full \n");
}
}
void display(){
for (i=0; i<=size-1; i++){
printf("\n %d \n", queue[i]);
}
}
int main(){
int choice;
printf("\n Enter the size of the queue: ");
scanf("%d", &size);
while(true){
printf("\n 1. enqueue\n");
printf("7. enqueue from front\n");
printf("2. dequeue\n");
printf("3. peek\n");
printf("4. display\n");
printf("5. check if queue is empty\n");printf("6. check if queue is full\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice){
case 1:
printf("\n Enter the element: ");
scanf("%d", &data);
enqueueR(data, size);
break;
case 2:dequeue();
break;
case 3:
peek();
break;
case 4:
display();
break;
case 5:
isEmpty();
break;
case 6:
isFull();
break;
case 7:
printf("\n Enter the element: ");
scanf("%d", &data);
enqueueF(data, size);
break;
default:
printf("Invalid choice\n");
}
}
return 0;
}
```

2. Write a menu driven program to implement Deques (both Input-restricted and Output-restricted) operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using linked list.

Input restricted:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
struct node {
int data;
struct node* next;
};
struct deque {
struct node* front;
struct node* rear;
int size;
int count;
};
struct node* createNode(int data) {
struct node* newNode = (struct node*)malloc(sizeof(struct node));
if (newNode != NULL) {
newNode->data = data;
newNode->next = NULL;
}
return newNode;
}
struct deque* createDeque(int size) {struct deque* dq = (struct deque*)malloc(sizeof(struct
deque));
if (dq != NULL) {
dq->front = NULL;dq->rear = NULL;
dq->size = size;
dq->count = 0;
}
return dq;
}
bool isFull(struct deque* dq) {
return dq->count == dq->size;
}
```

```
bool isEmpty(struct deque* dq) {
return dq->count == 0;
}
void enqueueRear(struct deque* dq, int data) {
if (isFull(dq)) {
printf("Deque overflow\n");
} else {
struct node* newNode = createNode(data);
if (newNode != NULL) {
if (isEmpty(dq)) {
dq->front = newNode;
dq->rear = newNode;
} else {
dq->rear->next = newNode;
dq->rear = newNode;
}
dq->count++;
printf("Enqueued element at the rear: %d\n", data);
} else {
printf("Memory allocation error\n");
}
}
}
void dequeueFront(struct deque* dq) {
if (isEmpty(dq)) {
printf("Deque underflow\n");
} else {
struct node* temp = dq->front;
dq->front = dq->front->next;
free(temp);
dq->count--;
printf("Element dequeued from the front\n");
}
}
void dequeueRear(struct deque* dq) {
if (isEmpty(dq)) {
printf("Deque underflow\n");
} else {
struct node* temp = dq->front;
struct node* prev = NULL;while (temp->next != NULL) {
prev = temp;
temp = temp->next;
}
if (prev != NULL) {
prev->next = NULL;
dq->rear = prev;
} else {
dq->front = NULL;
dq->rear = NULL;
}
free(temp);
```

```
dq->count--;
printf("Element dequeued from the rear\n");
}
}void display(struct deque* dq) {
if (isEmpty(dq)) {
printf("Deque is empty\n");
} else {
printf("Deque elements: ");
struct node* temp = dq->front;
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
}
void freeDeque(struct deque* dq) {
struct node* temp = dq->front;
while (temp != NULL) {
struct node* current = temp;
temp = temp->next;
free(current);
}
free(dq);
}
int main() {
int size, choice, data;
printf("Enter the size of the deque: ");
scanf("%d", &size);
struct deque* dq = createDeque(size);
while (true) {
printf("\nMenu:\n");
printf("1. Enqueue Rear\n");
printf("2. Dequeue Front\n");
printf("3. Dequeue Rear\n");
printf("4. Display\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);switch (choice) {
case 1:
printf("Enter element to enqueue at the rear: ");
scanf("%d", &data);
enqueueRear(dq, data);
break;
case 2:
dequeueFront(dq);
break;
case 3:
dequeueRear(dq);
break;
case 4:
```

```
display(dq);
break;
case 5:
freeDeque(dq);
printf("Exiting program...\n");
exit(0);
default:
printf("Invalid choice. Please try again.\n");
}
}
return 0;
}
```

Output restricted:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>struct node {
int data;
struct node* next;
};
struct deque {
struct node* front;
struct node* rear;
int size;
int count;
};
void* createNode(int data) {
struct node* newNode = (struct node*)malloc(sizeof(struct node));
if (newNode != NULL) {
newNode->data = data;
newNode->next = NULL;
}
return newNode;
}
void* createDeque(int size) {
struct deque* dq = (struct deque*)malloc(sizeof(struct deque));if (dq != NULL) {
dq->front = NULL;
dq->rear = NULL;
dq->size = size;
dq->count = 0;
}
return dq;
}
bool isFull(struct deque* dq) {
return dq->count == dq->size;
}
bool isEmpty(struct deque* dq) {
return dq->count == 0;
}
void enqueueFront(struct deque* dq, int data) {
if (isFull(dq)) {
printf("Deque overflow\n");
```

```
} else {
struct node* newNode = (struct node*)createNode(data);
if (newNode != NULL) {
if (isEmpty(dq)) {
dq->front = newNode;
dq->rear = newNode;
} else {
newNode->next = dq->front;
dq->front = newNode;
}
dq->count++;
printf("Enqueued element at the front: %d\n", data);
} else {
printf("Memory allocation error\n");
}
}
}
void enqueueRear(struct deque* dq, int data) {
if (isFull(dq)) {
printf("Deque overflow\n");
} else {
struct node* newNode = (struct node*)createNode(data);
if (newNode != NULL) {
if (isEmpty(dq)) {
dq->front = newNode;
dq->rear = newNode;
} else {
dq->rear->next = newNode;dq->rear = newNode;
}
dq->count++;
printf("Enqueued element at the rear: %d\n", data);
} else {
printf("Memory allocation error\n");
}
}}
void dequeueFront(struct deque* dq) {
if (isEmpty(dq)) {
printf("Deque underflow\n");
} else {
struct node* temp = dq->front;
dq->front = dq->front->next;
free(temp);
dq->count--;
printf("Element dequeued from the front\n");
}
}
void display(struct deque* dq) {
if (isEmpty(dq)) {
printf("Deque is empty\n");
} else {
printf("Deque elements: ");
```

```
struct node* temp = dq->front;
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
}
void freeDeque(struct deque* dq) {
struct node* temp = dq->front;
while (temp != NULL) {
struct node* current = temp;
temp = temp->next;
free(current);
}
free(dq);
}
int main() {
int size, choice, data;
printf("Enter the size of the deque: ");
scanf("%d", &size);
struct deque* dq = (struct deque*)createDeque(size);
while (true) {
printf("\nMenu:\n");
printf("1. Enqueue Front\n");
printf("2. Enqueue Rear\n");
printf("3. Dequeue Front\n");
printf("4. Display\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:printf("Enter element to enqueue at the front: ");
scanf("%d", &data);
enqueueFront(dq, data);
break;
case 2:
printf("Enter element to enqueue at the rear: ");
scanf("%d", &data);
enqueueRear(dq, data);break;
case 3:
dequeueFront(dq);
break;
case 4:
display(dq);
break;
case 5:
freeDeque(dq);
printf("Exiting program...\n");
exit(0);
default:
```

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

3. Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node* next;
};
struct CQueue {
struct Node* rear;
};
void* createNode(int data) {
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
if (newNode != NULL) {
newNode->data = data;
newNode->next = NULL;
}
return newNode;
}
void* createCQueue() {struct CQueue* cqueue = (struct CQueue*)malloc(sizeof(struct
CQueue));
if (cqueue != NULL) {
cqueue->rear = NULL;
}
return cqueue;
}
bool isEmpty(struct CQueue* cqueue) {
return cqueue->rear == NULL;
}
void enqueue(struct CQueue* cqueue, int data) {
struct Node* newNode = (struct Node*)createNode(data);
if (isEmpty(cqueue)) {
newNode->next = newNode;
cqueue->rear = newNode;
} else {
newNode->next = cqueue->rear->next;
cqueue->rear->next = newNode;
cqueue->rear = newNode;
}
printf("Enqueued element: %d\n", data);
}
void dequeue(struct CQueue* cqueue) {
if (isEmpty(cqueue)) {
printf("Queue underflow\n");
```

```
} else {struct Node* front = cqueue->rear->next;
if (front == cqueue->rear) {
free(front);
cqueue->rear = NULL;
} else {
cqueue->rear->next = front->next;
free(front);
}
printf("Dequeued element\n");
}
}
void peek(struct CQueue* cqueue) {
if (isEmpty(cqueue)) {
printf("Queue is empty\n");
}
else {
printf("Front element: %d\n", cqueue->rear->next->data);
}
}
void display(struct CQueue* cqueue) {
if (isEmpty(cqueue)) {
printf("Queue is empty\n");
} else {
struct Node* temp = cqueue->rear->next;printf("Queue elements: ");
do {
printf("%d ", temp->data);
temp = temp->next;
} while (temp != cqueue->rear->next);
printf("\n");
}
}
void freeCQueue(struct CQueue* cqueue) {
if (isEmpty(cqueue)) {
free(cqueue);
return;
}
struct Node* temp = cqueue->rear->next;
struct Node* nextNode;
do {
nextNode = temp->next;
free(temp);
temp = nextNode;
} while (temp != cqueue->rear->next);
free(cqueue);
}
int main() {
struct CQueue* cqueue = (struct CQueue*)createCQueue();
int choice, data;
while (true) {
printf("\nMenu:\n");
printf("1. Enqueue\n");
printf("2. Dequeue\n");
printf("3. Peek\n");
printf("4. Display\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter element to enqueue: ");
scanf("%d", &data);
enqueue(cqueue, data);
break;
case 2:
dequeue(cqueue);break;
case 3:
peek(cqueue);
break;
case 4:
display(cqueue);
break;
case 5:freeCQueue(cqueue);
printf("Exiting program...\n");
exit(0);
default:
printf("Invalid choice. Please try again.\n");
}
}
return 0;
}
```