

b) Write to implement singly linked list to simulate static queue operations.

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

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

```
struct Node {
```

```
    int data;
```

```
    struct Node *next;
```

```
};
```

```
struct Node *top = NULL;
```

```
struct Node *front = NULL;
```

```
struct Node *rear = NULL;
```

```
struct Node *createNode (int value)
```

```
{
```

```
    struct Node *newnode = (struct Node *)
```

```
    malloc (sizeof (struct Node));
```

```
    if (!newnode)
```

```
    {
```

```
        printf ("Memory allocation failed !\n");
```

```
        exit (0);
```

```
}
```

```
    newnode->data = value;
```

```
    newnode->next = NULL;
```

```
    return newnode;
```

```
}
```

```
void push (int value)
```

```
{
```

```
    struct Node *newnode = createNode (value);
```

```
newnode->next = top;
top = newnode;
stack.push(value);
```

y.

```
void pop()
{
    if (top == NULL)
        cout << "Stack is empty. nothing to
        pop." << endl + 7;
}
```

if (top->next == NULL)
 cout << "Stack is empty. nothing to
 pop." << endl + 7;

struct node \*temp = top;
printf("Node popped from the stack.
in", top->data);
top = top->next;
free(temp);
return;

y.

```
else
    void displayStack()
{
    struct node *temp = top;
    if (temp == NULL)
        cout << "Stack is empty." << endl + 7;
}
```

```
if (temp == NULL)
    cout << "Stack is empty." << endl + 7;
else
    cout << "Stack Top: " << temp->data;
    cout << endl + 7;
}
```

y.

```
prentf("Stack Top is %d\n", top);
while (temp != NULL)
{
}
```

Print ("got", temp->data);  
temp = temp->next;  
y

Print ("in");  
y.

void enqueue(Any value)

{  
struct node \* newnode = (newnode(value));  
if (front == rear == NULL)  
Front = rear = newnode;

else {

rear->next = newnode;  
rear = newnode;

Print ("good enqueue to the queue.\n",  
value);  
y,

void dequeue()

{  
if (front == NULL) {  
Print ("queue is empty. nothing to  
dequeue.\n");  
return ;  
y,

struct node \* temp = front;  
Print ("dequeue from the queue.  
\n", front->data);  
front = front->next;  
y.

```
if (front == null)
    rear = null;
freeListp;
```

```
y
```

```
void displayQueue()
```

```
{
```

```
struct node *temp = front;
```

```
if (temp == null)
    cout << "queue is empty." << endl;
```

```
else
    cout << "queue is empty." << endl;
```

```
y
print ("queue front is rear.");
```

```
else
    print ("good", temp->data);
    temp = temp->next;
```

```
y
print ("in");
y
```

```
cout << "displaying list"
    << endl;
```

```
if (rear != null)
```

```
    cout << "choice, value in : ";
    cout << endl;
```

```
while (1)
{
```

```
    print ("1. insert in front linked list"
        "2. insert at rear linked list"
        "3. display list"
        "4. search for a value"
        "5. delete a value"
        "6. exit");
    cout << endl;
```

```
cout << "choice, value in : ";
cout << endl;
```

```
if (choice == 1)
    insertFront();
else if (choice == 2)
    insertRear();
else if (choice == 3)
    displayList();
else if (choice == 4)
    search();
else if (choice == 5)
    deleteValue();
else if (choice == 6)
    exit(0);
```

pushf ("Enter your choice:");  
scanf ("%d", &choice);

switch (choice){

case 1:

```
    while(1){  
        printf ("In --- Stack menu...\n");  
        printf ("1. Push\n");  
        printf ("2. Pop\n");  
        printf ("3. Display stack\n");  
        printf ("4. Back to main menu");  
        scanf ("%d", &ch);  
        if(ch == 1){  
            pushf ("Enter value:");  
        } else if(ch == 2){  
            popf ();  
        } else if(ch == 3){  
            displayStack();  
        } else if(ch == 4){  
            break;  
        }  
    }  
}
```

case 2:

```
    pushf ("Enter value:");  
    scanf ("%d", &value);  
    pushf (value);  
    break;
```

case 3:

```
    displayStack();  
    break;
```

case 4:

```
    goto main-menu;  
}
```

break;

case 2:  
while (1) {  
 print ("in ... queue menu ->").  
 print ("1. enqueue \n");  
 print ("2. dequeue \n");  
 print ("3. display queue \n");  
 print ("4. public domain menu").  
 print ("5. enter your choice").  
 print ("6. exit \n");  
 scan (&"val", &ch);  
 if (ch == '1')  
 insert (val);  
 else if (ch == '2')  
 remove (val);  
 else if (ch == '3')  
 display (val);  
 else if (ch == '4')  
 display (menu);  
 else if (ch == '5')  
 break;  
}

case 1:

print ("enqueue value to  
enqueue:").  
scanf (&"val", &val);  
enqueue (val);

case 2:

dequeue();  
breake;

case 3:

displayqueue();  
breake;

case 4:

goto main - menu;  
default:

print ("invalid choice").  
try again \n");

break;

case 3:

```
print("empty program.\n");
```

default:

```
print("invalid choice. Try again.\n");
```

y

main-menu:

```
return 0;
```

y.

output:

-- singly linked list simulation --  
1. stack operation

2. queue operation

)

exit.

enter your choice: 1

-- stack -- menu

1. push

2. pop

3. display stack

4. back to main menu

enter your choice : 3  
stack is empty.

-- stack menu

1. push

2. pop