# Data Structure and Algorithm lab Experiment

# Queue implementation using linked list

Name: Om Ashish Mishra

Reg. No.: 16BCE0789

Slot: G2

**The Question:**

Create linked queue in such way it should accept input from user as both int and char where input is displayed as two linked queue one as int queue and another one char queue. For deletion it should follow the normal queue operation but it should delete both front elements.

Normal Queue Implementation using linked list

```
#include<stdio.h>

struct Node
{
   int item;
   struct Node*next;
}*front=NULL,*rear=NULL;

void insert(int);
void delete();
void display();

void main()
{
   int choice,value;
   printf("\n::Queue Implementation using linked list::\n");
   while(1)
   {
      printf("\n***MENU***\n");
      printf("\n1.Insert \n2.Delete \n3.Display \n4.Exit \n");
```

```
        printf("Enter your choice");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the values to be inserted ");
                    scanf("%d",&value);
                    insert(value);
                    break;
            case 2: delete();break;
            case 3: display();break;
            case 4: exit(0);
            default: printf("\nWrong selection!!! Please try again!!!\n");
        }
    }
}

void insert(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->item=value;
    newNode->next=NULL;
```

```
    if(front==NULL)
        front=rear=newNode;
    else
    {
        rear->next=newNode;
        rear=newNode;
    }
    printf("\n Insertion is Done!\n");
}

void delete()
{
    if(front==NULL)
        printf("\n Queue is Empty!!!\n");\
    else
    {
        struct Node *temp=front;
        front=front->next;
        printf("\nDeleted element: %d\n",temp->item);
        free(temp);
    }
}
```

```
void display()
{
    if(front==NULL)
    {
        printf("\n Queue is Empty!!!");
    }
    else
    {
        struct Node *temp=front;
        while(temp->next!=NULL)
        {
            printf("%d--->",temp->item);
            temp=temp->next;
        }
        printf("%d-->NULL\n",temp->item);
    }
}
```

**Queue implementation using two Linked List**

The Pseudo code:

- First we declare the required header files.
- Then we declare the size of the macro which is MAX 256 in this case.
- Then we create the 2 queues one for integer and other for alphabets.
- Then we write the required functions related to each case like insert, delete for integer queue and enqueue(insertion of elements), dequeue(delection of elements), init_queue(initialize the rear) and display is common to both.
- Then we call the main function for asking the user for the details he/she wants like on insertion we check for numbers and character and hence the value is stored in the queues accordingly.
- Then if the user presses 2 for delete then the program would simultaneously delete the elements from both queues.
- Then if the user presses 3 for display it would print simultaneously for the user's wanted position from both the queues.
- Then if the user presses 4 for exit, the program gets terminated.

The Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX 256


struct Node

{

   int item;

   struct Node*next;

}*front=NULL,*rear=NULL;


struct element {

   char *name;

   struct element *next;

};


void insert(int);

void delete();

void display();
```

```
struct element *tail;

void init_queue (void);
void enqueue (char *name);
int dequeue (char *name);
void print_queue (void);
void error (char *msg);

void main(int argc, char **argv)
{
   int choice,value,val;
   char buf [MAX];
   init_queue ();

   printf("\n::Queue Implementation using linked list::\n");
   while(1)
   {
      printf("\n***MENU***\n");
      printf("\n1.Insert \n2.Delete \n3.Display \n4.Exit \n");
      printf("Enter your choice");
      scanf("%d",&choice);
```

```
switch(choice)
{
    case 1: printf("Enter the values to be inserted is a number or character
1.Number and 2.Alphabet : ");
        scanf("%d",&val);
        if(val==1)
        {
            printf("Enter the values to be inserted ");
            scanf("%d",&value);
            insert(value);
        }
        else if(val==2)
        {
            if (fgets (buf, MAX, stdin) == NULL)
                break;
            printf ("Character :  ");
            if (fgets (buf, MAX, stdin) == NULL)
                break;
            int len = strlen (buf);
            if (buf [len - 1] == '\n')
                buf [len - 1] = '\0';
            enqueue (buf);
```

```
                }
                break;
            case 2: delete();
                if (dequeue (buf) != -1)
                printf ("%s dequeued\n", buf);
                break;
            case 3: display();
                break;
            case 4: exit(0);
            default: printf("\nWrong selection!!! Please try again!!!\n");
        }
    }
}

void insert(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->item=value;
    newNode->next=NULL;
    if(front==NULL)
        front=rear=newNode;
```

```
    else
    {
        rear->next=newNode;
        rear=newNode;
    }
    printf("\n Insertion is Done!\n");
}

void init_queue (void)
{
    tail = NULL;
}

void enqueue (char *name)
{
    struct element *ptr;
    char *cp;

    if ((ptr = (struct element *) malloc (sizeof (struct element))) == NULL)
        error ("malloc");
    if ((cp = (char *) malloc (strlen (name) + 1)) == NULL)
        error ("malloc");
```

```
    strcpy (cp, name);
    ptr -> name = cp;

    if (tail == NULL) {
        ptr -> next = ptr;
    }
    else
    {
        ptr -> next = tail -> next;
        tail -> next = ptr;
    }
    tail = ptr;
}

void delete()
{
    if(front==NULL)
        printf("\n Queue is Empty!!!\n");\
    else
    {
        struct Node *temp=front;
```

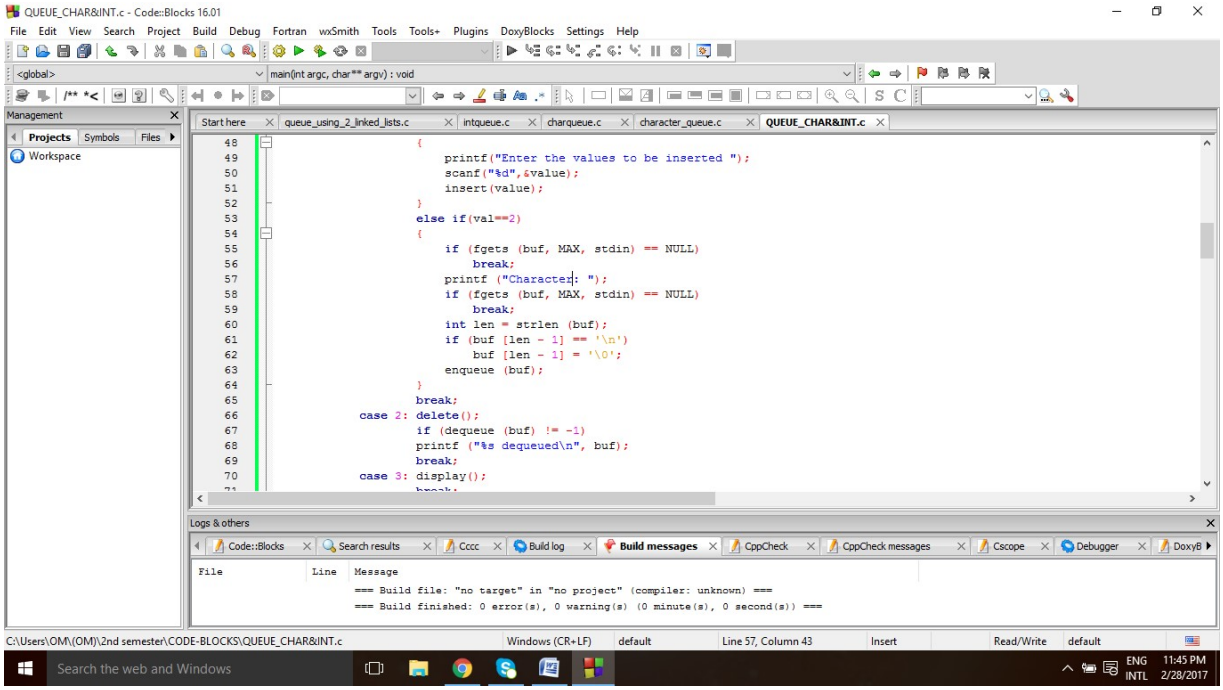
```
        front=front->next;
        printf("\nDeleted element: %d\n",temp->item);
        free(temp);
    }
}

int dequeue (char *name) // returns -1 on error
{
    struct element *ptr;
    char *cp;

    if (!tail) {
        fprintf (stderr, "Queue is empty\n");
        return -1;
    }
    // get the head
    ptr = tail -> next;
    cp = ptr -> name;

    if (ptr == tail)
        tail = NULL;
    else
```

```
        tail -> next = ptr -> next;
    free (ptr);
    strcpy (name, cp);
    free (cp);
    return 0;
}

void display(void)
{
    int n,i;
    printf("Enter the position wished by the user to see the queue details : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
    if(front==NULL)
    {
        printf("\n Queue is Empty!!!");
    }
    else
    {
        struct Node *temp=front;
        if(temp->next->next!=NULL)
```

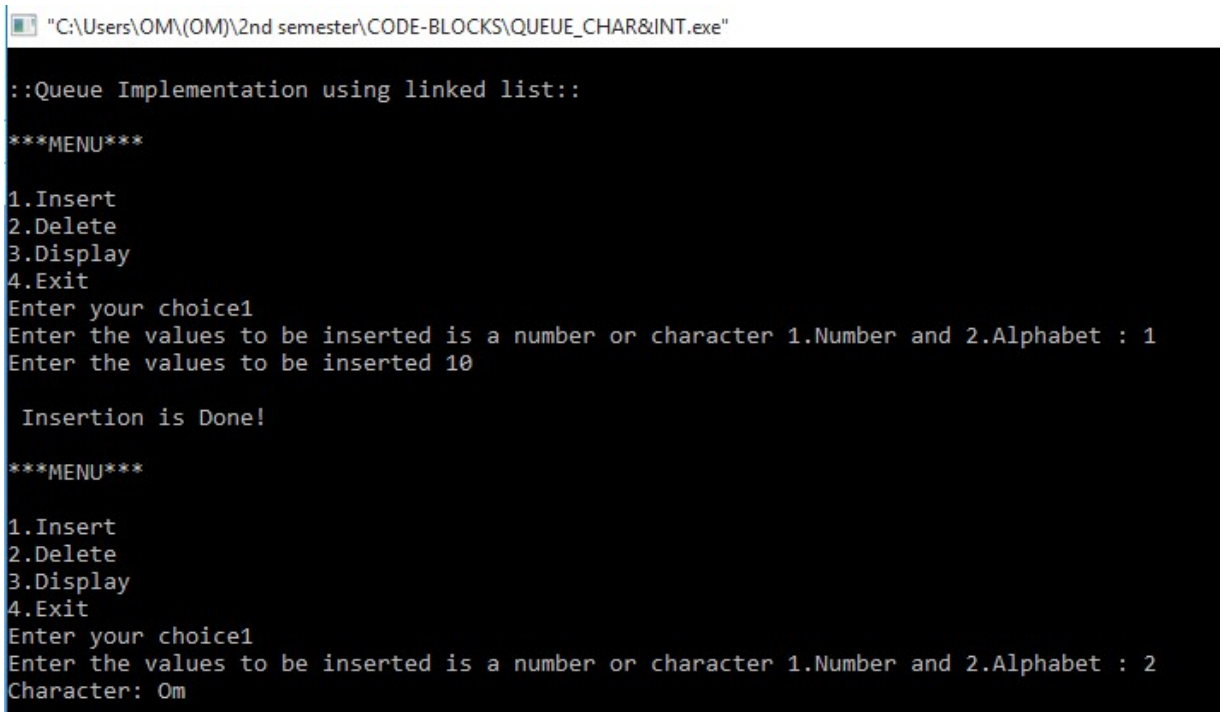
```
    {
        printf("%d--->",temp->item);
        temp=temp->next;
    }
    printf("%d-->NULL\n",temp->item);
}

struct element *ptr, *head;

if (!tail)
{
    fprintf (stderr, "Queue is empty\n");
    return;
}

printf ("Queue: \n");

// get the head
head = ptr = tail -> next;
int i = 1;

if(ptr!=head)
```

```
    {
        printf ("%d. %s\n", i, ptr -> name);
        ptr = ptr -> next;
        i++;
    }
    }
}

void error (char *msg)
{
    perror (msg);
    exit (1);
}
```

## The Output:




## Insertion



```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\QUEUE_CHAR&INT.exe"

::Queue Implementation using linked list::

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice1
Enter the values to be inserted is a number or character 1.Number and 2.Alphabet : 1
Enter the values to be inserted 10

 Insertion is Done!

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice1
Enter the values to be inserted is a number or character 1.Number and 2.Alphabet : 2
Character: Om
```

```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\QUEUE_CHAR&INT.exe"
***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice1
Enter the values to be inserted is a number or character 1.Number and 2.Alphabet : 1
Enter the values to be inserted 20

 Insertion is Done!

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice1
Enter the values to be inserted is a number or character 1.Number and 2.Alphabet : 2
Character: World
```

Deletion:

```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\QUEUE_CHAR&INT.exe"

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice2

Deleted element: 10
Om dequeued

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice2

Deleted element: 20
World dequeued
```

Display and exit :

```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\QUEUE_CHAR&INT.exe"
***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice3
Enter the position wished by the user to see the queue details : 1

 Queue is Empty!!!Queue is empty

***MENU***

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice4

Process returned 0 (0x0)   execution time : 332.923 s
Press any key to continue.
```

Thank You