

**19BIT0292**

**Bhaumik Tandan**

**LAB SHEET**

**DATA STRUCTURES**

**AND**

**ALGORITHMS**

**LABORATORY**

CSE2011

L57+L58

**Q1) Implementation of stack using array**

**Push(), Pop(), Display()**

**stack.h**

**CODE**

#include "./stack\_header/varibles\_decalred.h"//it also contains header files

#include "./stack\_header/stack\_functions.h"

#include "./stack\_header/push\_type.h"

#pragma once//restrict double import

#define push(st,a) \_Generic(a, int: pushi\_\_19BIT0292, char\*: pushs\_\_19BIT0292,double: pushf\_\_19BIT0292,char:pushc\_\_19BIT0292,float:pushf\_\_19BIT0292)(st,a)//char and int will be treated similarly

void s\_in(stack \*s)

{

s->t\_\_19BIT0292=-1;

s->stack\_\_19BIT0292=0;

s->d\_type\_\_19BIT0292=0;

}

void menu(stack \*st)

{

void\* (\* fp[3])(stack \*);

//0 push

//1 pop

//2 top

//3 display whole stack

fp[0]=&pop;

fp[1]=&top;

fp[2]=&display;

printf("\n\n\n1)Push\n2)Pop\n3)Top\n4)Display\n5)Exit\n");

printf("\nEnter your choice: ");

int c;

scanf("%d",&c);

if(c==1)

{

printf("\n\nEnter that you to push in the stack: ");

char s[21];//this will get destroyed after function is finished it also has null

scanf("%s",s);

int a=atoi(s);//convert string to int

float f=atof(s);

if((a!=0 || strcmp("0",s)==0)&& f==a)

{

push(st,a);

return menu(st);

}

if(f!=0)

{

push(st,f);

return menu(st);

}

if(strlen(s)>1){

push(st,s);

}

else

push(st,s[0]);

return menu(st);

}

else if(c==5)

return;

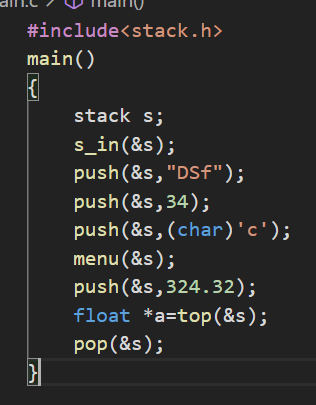
fp[c-2](st);

return menu(st);

}

**main.c**

**CODE**

#include<stack.h>

main()

{

stack s;

s\_in(&s);

push(&s,"DSf");

push(&s,34);

push(&s,(char)'c');

menu(&s);

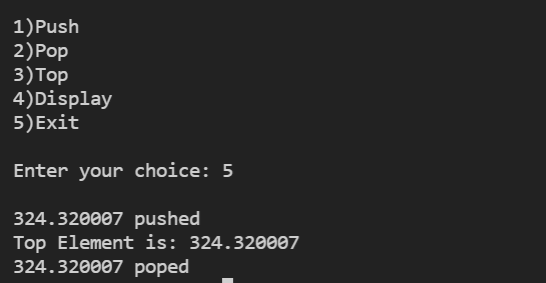
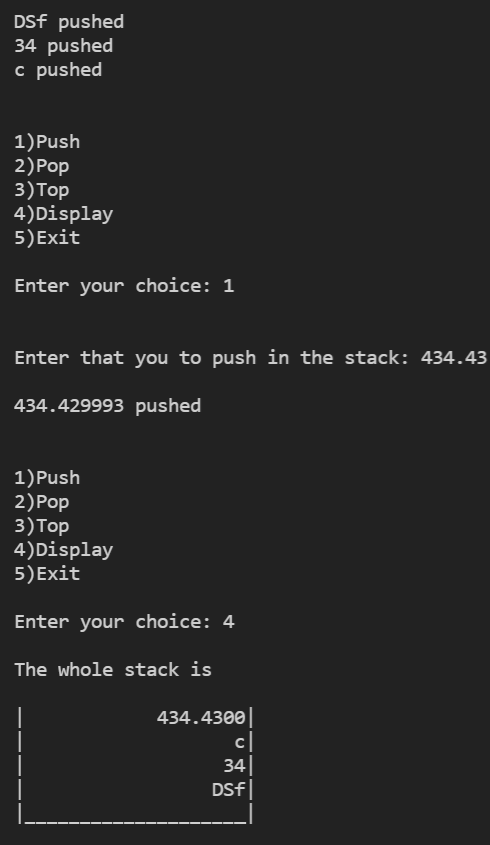
push(&s,324.32);

float \*a=top(&s);

pop(&s);

}

**OUTPUT**

****

# [**CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE**](https://github.com/Bhaumik-Tandan/Generic_Stack_using_c)

**Q2) Conversion of infix expressions to postfix expressions using stack A+B/C – D +(E^F)**

**CODE**

#include <stdio.h>

#include <string.h>

char stack[50];

int top = -1;

void push(char c)

{

top++;

stack[top] = c;

}

char pop()

{

char c;

if (top == -1)

return -1;

else

{

c = stack[top];

top--;

return c;

}

}

int priority(char x)

{

if (x == '(')

return 0;

else if (x == '+' || x == '-')

return 1;

else if (x == '\*' || x == '/')

return 2;

else if (x == '^')

return 3;

}

main()

{

char exp[100];

char \*p, x;

printf("Enter the expression :: ");

scanf("%s", exp);

p = exp;

while (\*p != '\0')

{

if (isalnum(\*p))

printf("%c", \*p);

else if (\*p == '(')

push(\*p);

else if (\*p == ')')

{

while ((x = pop()) != '(')

printf("%c", x);

}

else

{

while (priority(stack[top]) >= priority(\*p))

printf("%c", pop());

push(\*p);

}

p++;

}

while (top != -1)

{

printf("%c", pop());

}

}

**OUTPUT**

****

****

# [**CLICK HERE FOR GITHUB LINK**](https://github.com/Bhaumik-Tandan/C_codes/blob/master/infix_post.c)

**Q3) Evaluation of postfix expression using stack**

**CODE**

#include <stdio.h>

#include <math.h>

int s[100];

int top = -1;

push(int e)

{

s[++top] = e;

}

int pop()

{

return (s[top--]);

}

main()

{

int i = 0, v1, v2;

char c[100];

scanf("%s", c);

while (c[i] != '\0')

{

if (isdigit(c[i]))

{

push(c[i] - 48);

}

else

{

v1 = pop();

v2 = pop();

switch (c[i])

{

case '+':

push(v2 + v1);

break;

case '-':

push(v2 - v1);

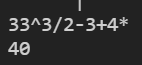
break;

case '\*':

push(v2 \* v1);

break;

case '/':

push(v2 / v1);

break;

}

}

i++;

}

printf("%d", pop());

} [**CLICK HERE FOR GITHUB LINK**](https://github.com/Bhaumik-Tandan/C_codes/blob/master/post_to_infix.c)

**Q4)** Implement Queue and realize various operations to be carried out on it.

**queue.h**

**CODE**

#include "./queue\_header/varibles\_decalred.h"//it also contains header files

#include "./queue\_header/queue\_functions.h"

#include "./queue\_header/enqueue\_type.h"

#pragma once//restrict double import

#define enqueue(s,a) \_Generic(a, int: enqueuei\_\_19BIT0292, char\*: enqueues\_\_19BIT0292,double: enqueuef\_\_19BIT0292,char:enqueuec\_\_19BIT0292,float:enqueuef\_\_19BIT0292)(s,a)

void q\_in(queue \*q)

{

q->r\_\_19BIT0292=-1;

q->queue\_\_19BIT0292=0;

q->d\_type\_\_19BIT0292=0;

}

void menu(queue \*q)

{

void\* (\* fp[4])(queue\*);

fp[0]=&denqueue;

fp[1]=&front;

fp[2]=&rear;

fp[3]=&display;

printf("\n\n\n1)Enqueue\n2)Dequeue\n3)Front\n4)Rear\n5)Display\n6)Exit\n");

printf("\nEnter your choice: ");

int c;

scanf("%d",&c);

if(c==1)

{

printf("\n\nEnter that you to enqueue in the stack: ");

char s[21];

scanf("%s",s);

int a=atoi(s);//convert string to int

float f=atof(s);

if((a!=0 || strcmp("0",s)==0)&& f==a)

{

enqueue(q,a);

return menu(q);

}

if(f!=0)

{

enqueue(q,f);

return menu(q);

}

if(strlen(s)>1)

enqueue(q,s);

else

enqueue(q,s[0]);

return menu(q);

}

else if(c==6)

return;

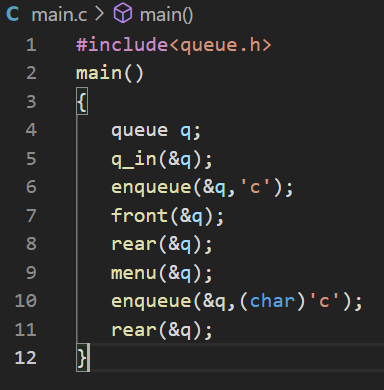
fp[c-2](q);

return menu(q);

}

**main.c**

**CODE**

#include<queue.h>

main()

{

queue q;

q\_in(&q);

enqueue(&q,'c');

front(&q);

rear(&q);

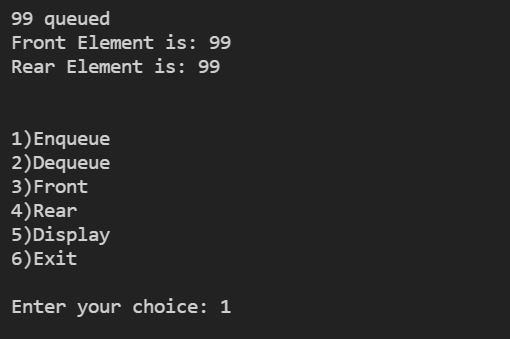
menu(&q);

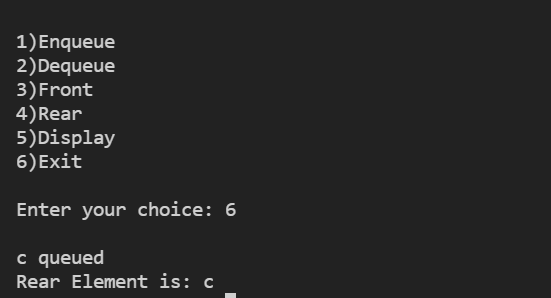
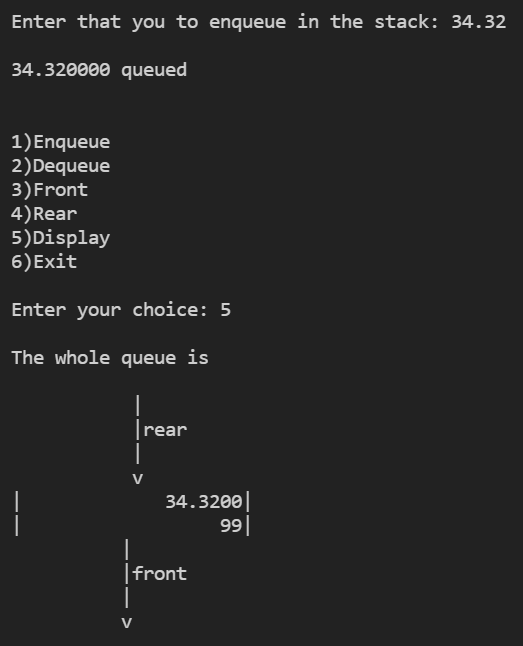
enqueue(&q,(char)'c');

rear(&q);

}

**OUTPUT**





# [**CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE**](https://github.com/Bhaumik-Tandan/Generic_queue_in_c)

**Q5) Implementation of circular queue.**

**CODE**

#include<stdio.h>

#define s 5

int ar[s],f=0,r=-1,e;

add(int a)

{

if(e==s){

printf("\nQueue Full");

return 1;

}

r=(r+1)%s;

ar[r]=a;

e++;

printf("\n%d added",a);

}

del()

{

if(e==0)

{

printf("\nQueue Empty");

return;

}

e--;

int t=f;

f=(f+1)%s;

return ar[t];

}

disp()

{

if(e==0)

{

printf("\nQueue Empty");

return;

}

printf("\nFront ---> ");

int i=f;

while (i!=r)

{

printf("%d ",ar[i]);

i=(i+1)%s;

}

printf("%d ---> Rear",ar[i]);

}

main()

{

add(3);

add(23);

add(94);

add(232);

add(4);

add(231);

disp();

printf("\n %d deleted",del());

add(299);

disp();

printf("\n %d deleted",del());

printf("\n %d deleted",del());

printf("\n %d deleted",del());

printf("\n %d deleted",del());

printf("\n %d deleted",del());

disp();

add(3);

add(23);

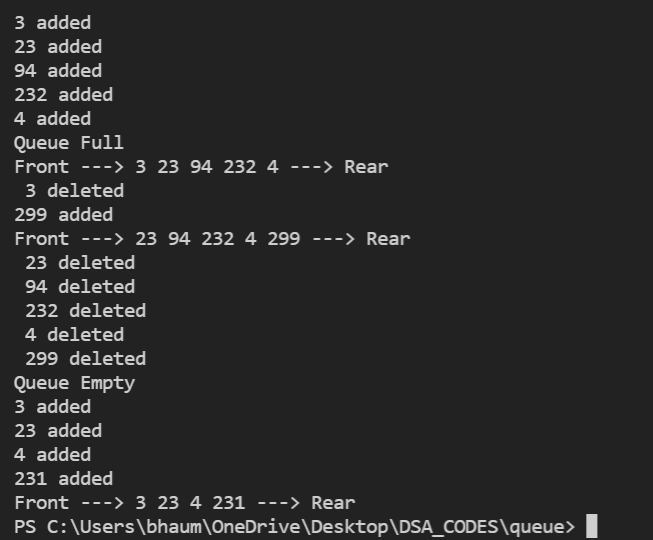
add(4);

add(231);

disp();

}

**OUTPUT**

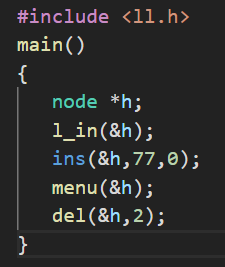


[**CLICK HERE FOR GITHUB LINK**](https://github.com/Bhaumik-Tandan/DSA_CODES/blob/master/queue/circular_queue.c)

**Q6) . Implementation of singly linked list**

**main.c**

**CODE**

#include <ll.h>

main()

{

node \*h;

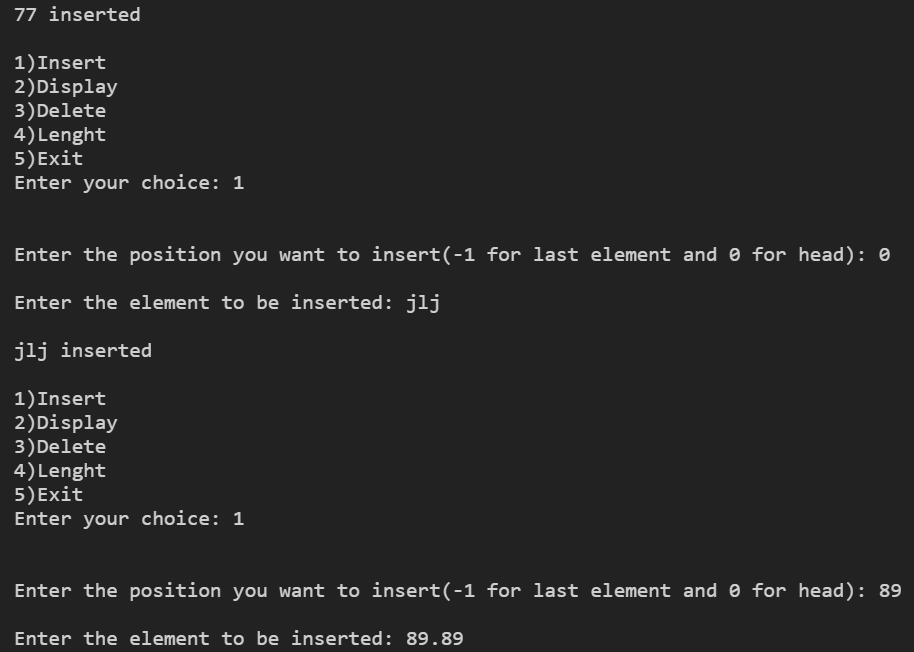
l\_in(&h);

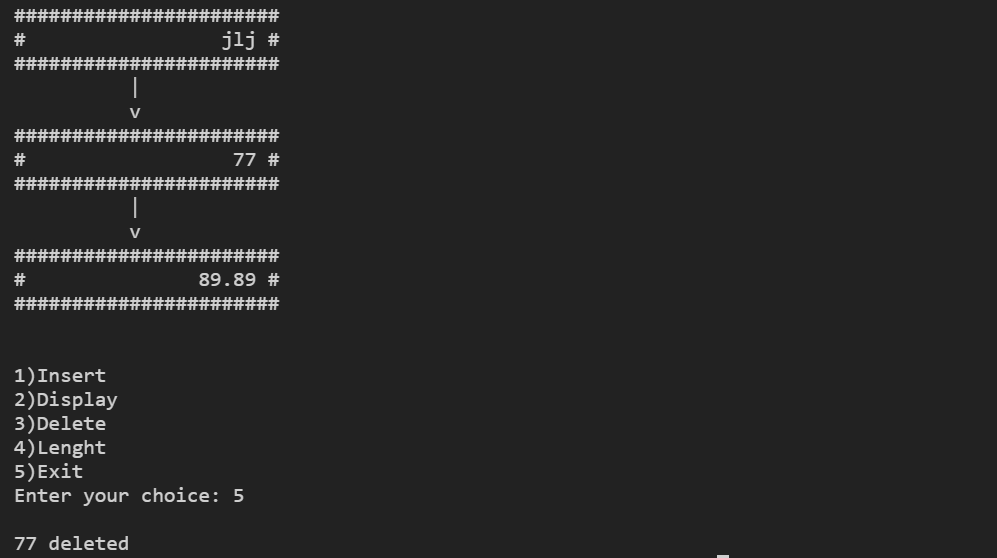
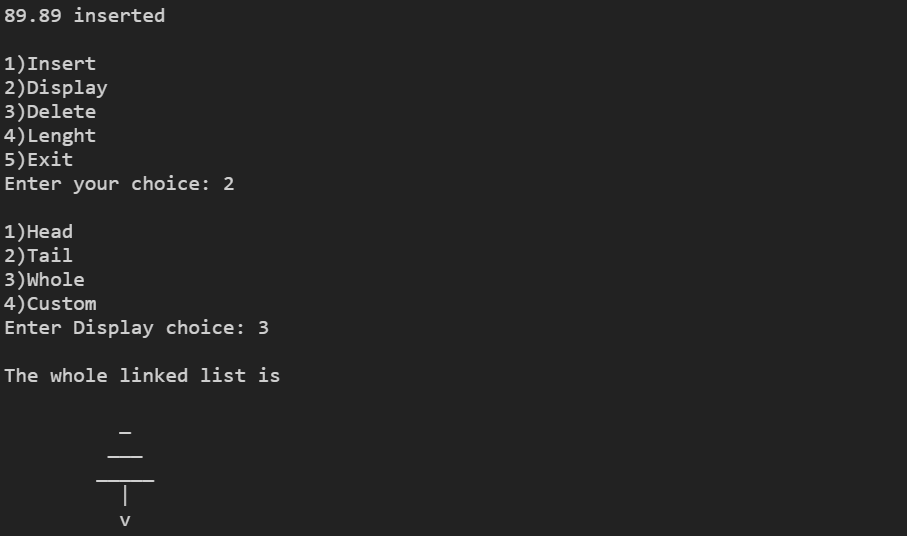
ins(&h,77,0);

menu(&h);

del(&h,2);

}

**OUTPUT**



# [**CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE**](https://github.com/Bhaumik-Tandan/Generic_linked_list_in_c)

**Q7) Implementation of stack using linked list**

**CODE**

#include <stdio.h>

struct node

{

int d;

struct node \*n;

} \* h;

typedef struct node node;

push(int d)

{

node \*t = malloc(sizeof(node));

t->d = d;

t->n = h;

h=t;

printf("\n%d pushed",d);

}

pop()

{

if (!h)

{

printf("\nStack Empty");

return;

}

int a = h->d;

h = h->n;

return a;

}

disp()

{

if (!h)

{

printf("\nStack Empty");

return;

}

printf("\nStack is: ");

node \*p = h;

while (p)

{

printf("%d ", p->d);

p = p->n;

}

}

main()

{

push(3);

push(23);

push(94);

push(232);

push(4);

push(231);

disp();

printf("\n %d poped",pop());

push(299);

disp();

printf("\n %d poped",pop());

printf("\n %d poped",pop());

printf("\n %d poped",pop());

printf("\n %d poped",pop());

printf("\n %d poped",pop());

disp();

push(3);

push(23);

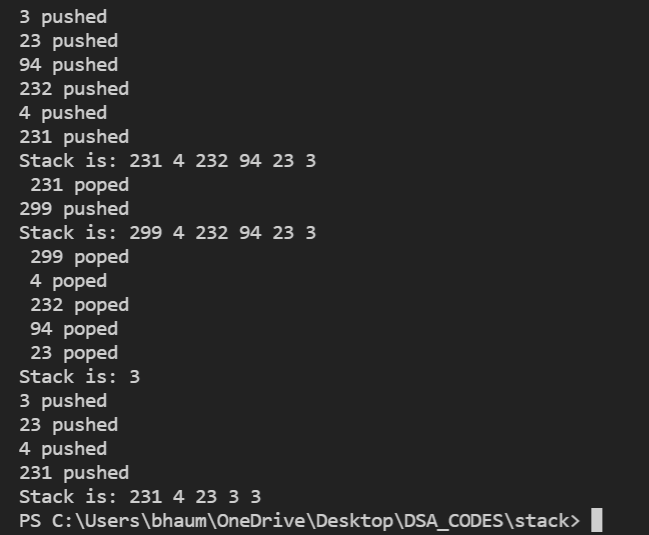
push(4);

push(231);

disp();

}

**OUTPUT**



# [CLICK HERE FOR GITHUB LINK](https://github.com/Bhaumik-Tandan/DSA_CODES/blob/master/stack/linked_list.c)

**Q8) Implementation of queue using linked list**

**CODE**

#include <stdio.h>

struct node

{

int d;

struct node \*n;

} \* h;

typedef struct node node;

enqueue(int d)

{

node \*t = malloc(sizeof(node));

t->d = d;

t->n = 0;

printf("\n%d enqueueed",d);

if (!h)

{

h = t;

return;

}

node \*p = h;

while (p->n)

p = p->n;

p->n = t;

}

dequeue()

{

if (!h)

{

printf("\nQueue Empty");

return;

}

int a = h->d;

h = h->n;

return a;

}

disp()

{

if (!h)

{

printf("\nQueue Empty");

return;

}

printf("\nQueue is: ");

node \*p = h;

while (p)

{

printf("%d ", p->d);

p = p->n;

}

}

main()

{

enqueue(3);

enqueue(23);

enqueue(94);

enqueue(232);

enqueue(4);

enqueue(231);

disp();

printf("\n %d dequeued",dequeue());

enqueue(299);

disp();

printf("\n %d dequeued",dequeue());

printf("\n %d dequeued",dequeue());

printf("\n %d dequeued",dequeue());

printf("\n %d dequeued",dequeue());

printf("\n %d dequeued",dequeue());

disp();

enqueue(3);

enqueue(23);

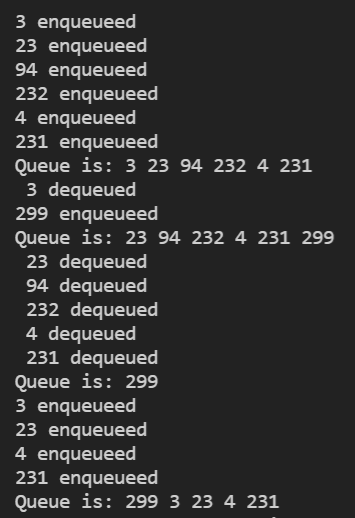
enqueue(4);

enqueue(231);

disp();

}

**OUTPUT**



# [CLICK HERE FOR GITHUB LINK](https://github.com/Bhaumik-Tandan/DSA_CODES/blob/master/queue/queue_using_ll.c)