store
67

A> write a program to simulate the working of queue

Pseudo code :-

insert :

1) if rear = max - 1
p('queue is full')
return

2) if rear = -1 , front = -1
rear = 0
front = 0

3 queue [rear] = element

4. return

Delete

1. if rear = front = -1
printf 'q is empty'
return

2. element = queue (front)

3. if front = rear
front = -1
rear = -1
else
front = front + 1

return (element)

display:

1. if rear = -1 & front = -1
   p ('q is empty')
   return.

2. print
   for (i ← front to rear)
   p(q[i])
   end for

3. return.

```
#include <stdio.h>
#include<stdlib.h>
int rear=-1, front=-1, queue[30];
#define max 20

int q[max];
int front = -1, rear = -1;

void ins(int)
int del();
void display();
int i;
void main()
{
int ch, item, x;
char a;
printf("queue implementation");
printf("1. insert\n");
printf("2. delete\n");
printf("3. display\n");
printf("4: exit");

do {
printf("\n\t enter your choice:");
scanf("%d", &ch);
switch(ch)
{
case 1: printf("enter element to insert:\n");
        Scanf("%d", &item);
        ins(item); if(x==1) printf("Queue is full");
        break;
case 2: x = del();
        printf("the element deleted from queue is");
        %d", x); break; if(x==1) printf("q is empty");
```

```
case 3:  display ();
         break;

case 4:  x = isempty ();
         if (x == 1)
         printf ("queue is empty")
         else
         printf ("que is not empty");
         break;

case 5:  x = isfull ();
         if (x == 1)
         print
```

```
case 4: exit (0); break;
default: printf ("invalid choice \n");
}
printf ("\n do u want to continue y/n: ");
scanf ("%c", &a);
}
while (a == 'y') || (a == 'Y'));
getch ();
}
```

```
Void ins (int x)
{
if (rear == mar - 1)
printf ("queue is overflow/n ");
else if (rear == -1)
 {
 front = 0, rear = 0;
    q [rear ] = x;
   }
else
  {
   rear ++ ;
  q [rear ] = x;
   }
}
int del ()
{
int x;
if (front == -1)
printf ("queue is underflow \n" )
 else if (front == rear )
{
 x = q [front ];
 front = -1;
 rear = -1;
 return (x);
}
 else
 {
  x = q [front ];
  front ++;
   return (x);
  }
 }
```

```
Void display ()
{
int i.
if (rear == -1)
 printf ("\n queue is Empty ");
 else
{
for (i= front; i <= rear ; i++)
 printf ("%d\t", q[i]);
}
}
}
```

Output

Queue implementation:

1: insert
2: delete
3: display
4: exit.

1

enter element to insert

11

1

enter element to insert

22

1

enter element to insert

33

1

enter element to insert

1

44

Stack ↑

store
67

1
stack overflow
2
element 11 deleted
2
element 22 deleted
2
element 33 deleted
2
element 44 deleted
2
underflow
1
enter element
10
3
10
4

```
#include <stdbool.h>
#include <string.h>

bool isValid(char *s){
int n = strlen(s);
char stack[n]
int top = -1;
int i = 0;
while(i<n)
{
char c = s[i];
if(c=='(' || c=='{' || c=='[')
    stack[++top] = c;
else
{
if(top==-1)
    return false;
char topchar = stack[top--];
if((c==')' && topchar!='(') ||
    (c=='}' && topchar!='{') ||
    (c==']' && topchar!='['))
    return false;
```

( )

true

( ) [ ] { }

true

[ ]

false

([ ]) }

true

15/10/24

Write a program to simulate circular queue in data structure using c language.

Insert.

```
if (f==0 && rear == size-1) || (rear == (front-1) %.size-1))
  Pf ("queue is full \n");
  return
else if (f==-1)
  f=0 ; r= 0;
  q[r] = value;
else if (r== size-1 && front!=0)
   { r= 0;
     q[r] = value;
    }
else {
    r++ ;
    q[r] = value ; }
}
```

del

```
if (f==-1)
  Pf ("queue is empty");
   return;
Pf ( deleted , que[front]);
 q(front) =-1;
 if (f== r)
   f=r=-1;
else if (front == size-1) {
  front= 0;
 }
else { front++ ; } }
```

```
display (){
if (front==-1){
   Pf ("queue is empty\n");
    return;
}
Pf ("queue elements are ").
 if (r >= f)
    for (int i=f ; i<=rear ; i++)
      Pf (q[i]))
 else
    for (i=front ; i<size ; i++)
        Pf
```

22/10/24

Write program to simulate circular queue.

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

int queue[size];
int front = -1;
int rear = -1;
Void enter (int value) {
if ((front == 0 && rear == size-1) || (rear == (front -1) %. (size -1)))
{ printf ("queue is full \n");
   return; }
else if (front == -1) {
   front == rear = 0;
   queue [rear] = value;
  }
else {
  rear + +;
  queue [rear] = value;
 }
printf ("Inserted %.d \n", value);
 }
Void del() {
   if (front = = -1) {
    printf ("queue is empty \n");
    return;
}
printf ("deleted %. d \n", queue [front]);
 queue [front] = -1;
if (front == rear) {
      front = = rear = -1;
}
```

```
else if (front == size - 1 ) {
front = 0;
}
else {
    front ++;
}
}
Void display () {
  if (front == -1 ) {
     printf ("queue is empty \n");
      return;
    }
printf ("queue elements are : ");
 if (rear >= front ) {
   for (int i  = front ; i <= rear ; i++ )
      printf ("%d ", queue[i] );
    }
else  {
    for (int i = front ; i<size ; i++ )
       printf ("%d", queue[i] );
    for (int i = 0 ; i <= rear ; i++ )
       printf ("%d ", queue [i] );
   }
  printf ("\n");
}

int main () {
int choice, value ;
while (1) {
   printf ("\n1. insert \n2. delete \n3. display \n4. Exit");
   printf ("enter your choice :");
   Scanf ("%d", &choice );
```

```
Switch ( choice ) {
 case 1 :
    printf("Enter the value to insert : ");
    Scanf ( "%d", &value );
    enter(value ) ;
    break;
case 2 :
      del ();
      break;
case 3 :
      display ();
      break;
case 4 :
      exit (0);
default :
      printf ("invalid choice \n");
  }
}
return 0;
 } .
```

Output

```
1. insert
2. delete
3. display
4. delete
enter your choice : 1
enter value to insert 1
inserted 1.
```

1. insert
2 delete
3. display
4 delete
enter your choice : 1
enter value to insert 2
inserted 2

1 insert
2 delete
3 display
4 delete
enter your choice : 1
enter value to insert 3
inserted 3
queue is full

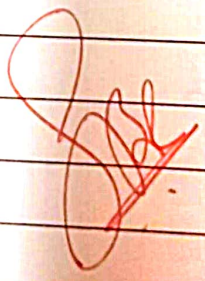
1 insert
2 delete
3 display
4 delete
enter your choice : 3
queue elements are : 123

1 insert
2 delete
3 display
4 delete
enter your choice : 2
deleted 1

1 insert
2 delete
3 display
4 exit
enter your choice 2
deleted 2

1 insert
2 delete
3 display
4 exit
enter your choice 2
deleted 3
queue is empty

1 insert
2 delete
3 display
4. exit
enter your choice 4.



store
67 22/10/2024

{ First unique character in a string }

```
int firstuniqchar(char *s) {
int freq[26]={0};
for(int i=0; s[i]!='\0'; i++){
    freq[s[i]-'a']++;
}
for(int i=0; s[i]!='\0'; i++){
   if(freq[s[i]-'a']==1){
        return i;
    }
}
    return -1;
}.
```

case 1

s = "leetcode"

output

0

case 2

s = "loveleetcode"

output

2.

Case 3

s = "aabb"

output

-1.

22/10/2024