***ASSIGNMENT-3***

***1. WAP to implement Queue.  
 a. Insert an element into the queue.  
 b. Delete an element from the queue.  
 c. Display the contents of the queue.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 5

struct queue{

int arr[SIZE];

int rear,front;

}s;

void enqueue(int data);

int dequeue();

void display();

int main()

{

int ch,data;

s.front = 0;

s.rear = -1;

while(1)

{

printf("Enter your choice what you want to do\n");

printf("1.Insert an element in queue\n");

printf("2.Delete an element \n");

printf("3.Display content of queue\n");

printf("4.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter element you want to insert\n");

scanf("%d",&data);

enqueue(data);

break;

case 2:

printf("element which is deleted is : %d\n\n",dequeue());

break;

case 3:

display();

break;

case 4:

exit(1);

}

if(s.front>s.rear)

{

s.front = 0;

s.rear = -1;

}

}

return 0;

}

void enqueue(int data)

{

if(s.rear == SIZE -1)

printf("Queue overflow\n");

else {

s.arr[++(s.rear)] = data;

}

}

int dequeue(){

if(s.front>s.rear){

printf("Queue underflow\n");

return -100;

}

else

return s.arr[(s.front)++];

}

void display()

{

int i;

if(s.front>s.rear)

printf("No element in the queue\n");

else

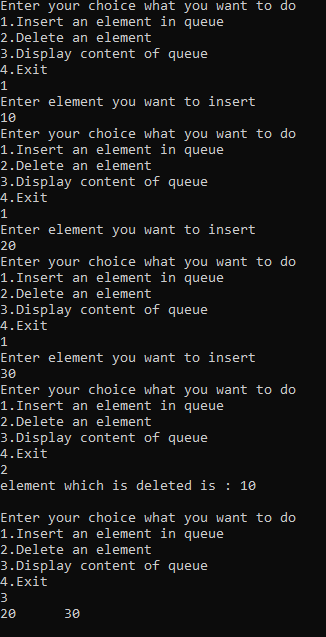
for(i=s.front;i<=s.rear;i++)

printf("%d\t",s.arr[i]);

printf("\n\n");

}

**OUTPUT:**



***2. WAP to implement Circular Queue.  
 a. Insert an element into the circular queue.  
 b. Delete an element from the circular queue.  
 c. Display the contents of the circular queue.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 50

struct queue{

int arr[SIZE];

int front,rear;

}s;

int cnt=0;

void enqueue(int data);

int dequeue();

void display();

int main()

{

s.front = 0;

s.rear = -1;

int data,ch;

while(1)

{

printf("Enter your choice what you want to do\n");

printf("1.Insert an element in Queue\n");

printf("2.Delete element from Queue\n");

printf("3.Display the Queue\n");

printf("4.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter element you want to insert\n");

scanf("%d",&data);

enqueue(data);

break;

case 2:

printf("Elememt which is deleted : %d\n\n",dequeue());

break;

case 3:

display();

break;

case 4:

exit(1);

}

}

return 0;

}

void enqueue(int data)

{

if(((s.rear+1)%SIZE == s.front)&& cnt>0)

printf("Queue overflow\n");

else{

s.rear = (s.rear+1)%SIZE;

s.arr[s.rear] = data;

++cnt ;

}

}

int dequeue()

{

if((((s.rear+1)%SIZE == s.front)&& cnt==0)||(s.rear == -1))

{

printf("Queue underflow\n");

return -1000;

}

else

{

int tmp;

tmp = s.arr[s.front];

s.front = (s.front+1)%SIZE;

--cnt;

return tmp;

}

}

void display()

{

if((((s.rear+1)%SIZE == s.front)&& cnt==0)||(s.rear == -1))

printf("Queue is empty\n");

else

{

int i;

int j = s.front;

for(i=0;i<cnt;i++)

{

printf("%d\t",s.arr[j]);

j = (j+1)%SIZE;

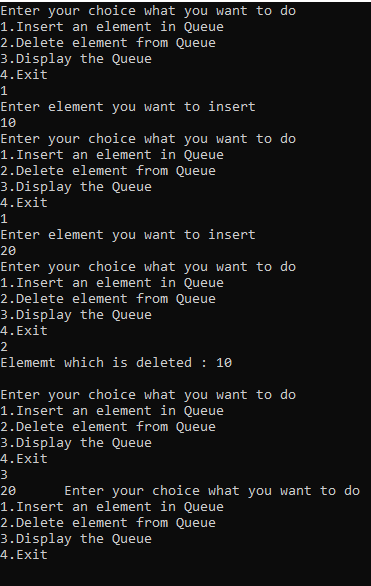
}

}

}

***OUTPUT :***

***/\*Whole output can’t be displayed . So, I am only displaying general queue output. But you can check it by execute it.\*/***



***3. WAP to implement Double Ended Queue.  
 a. Insert an element into the double ended queue.  
b. Delete an element from the double ended queue.  
c. Display the contents of the double ended queue.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 6

struct queue{

int arr[SIZE];

int front,rear;

}s;

int cnt = 0;

void enqueuefront(int data);

void enqueuerear(int data);

int dequeuerear();

int dequeuefront();

void display();

int main()

{

s.front = SIZE;

s.rear = -1;

int ch,data;

while(1)

{

printf("Enter your choice what you want to do:\n");

printf("1.insert an element at front\n");

printf("2.insert an element at rear\n");

printf("3.delete an element at front\n");

printf("4.delete an element at rear\n");

printf("5.Display queue\n");

printf("6.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter number you want to insert\n");

scanf("%d",&data);

enqueuefront(data);

break;

case 2:

printf("Enter number you want to insert\n");

scanf("%d",&data);

enqueuerear(data);

break;

case 3:

printf("number which is deleted is : %d\n\n",dequeuefront());

break;

case 4:

printf("number which is deleted is : %d\n\n",dequeuerear());

break;

case 5:

display();

break;

case 6:

exit(1);

}

}

return 0;

}

void enqueuefront(int data)

{

if(s.front - s.rear == 1)

printf("Queue overflow\n");

else{

cnt++;

s.arr[--(s.front)] = data;

}

}

void enqueuerear(int data)

{

if(s.front - s.rear ==1 )

printf("Queue overflow\n");

else{

cnt++;

s.arr[++(s.rear)] = data;

}

}

int dequeuefront()

{

if(s.front == SIZE){

printf("Queue underflow\n");

return INT\_MAX;

}

else

{

--cnt;

return s.arr[(s.front)++];

}}

int dequeuerear()

{

if(s.rear == -1){

printf("Queue underflow\n");

return INT\_MAX;

}

else{

--cnt;

return s.arr[(s.rear)--];

}

}

void display()

{

int i,j = s.front;

for(i=0;i<cnt;i++)

{

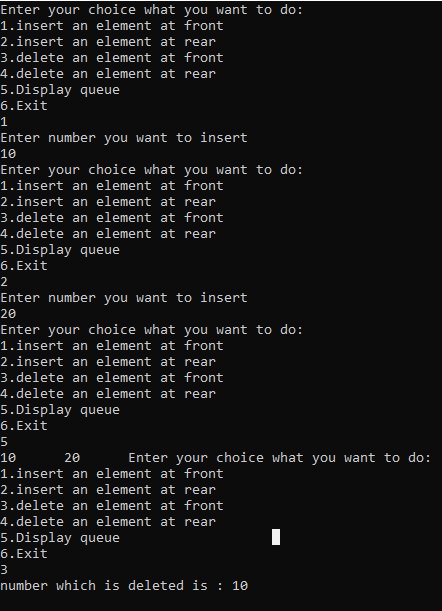
printf("%d\t",s.arr[j]);

j = (j+1)%SIZE;

}

}

***OUTPUT:***



***4. WAP to implement Priority Queue.  
 a. Insert an element into the priority queue.  
 b. Delete an element from the priority queue.  
 c. Display the contents of the priority queue.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 20

struct queue{

int item,pr;

}s[SIZE];

int top=-1;

void enqueue(int data,int p);

int dequeue();

int highestpr();

void display();

int main()

{

int data,ch,pr;

while(1)

{

printf("Enter your choice what you want to do\n");

printf("1.Insert an element\n");

printf("2.Delete element \n");

printf("3.get highest priority element\n");

printf("4.Display queue\n");

printf("5.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter elememt you want to insert and priority of element\n");

scanf("%d%d",&data,&pr);

enqueue(data,pr);

break;

case 2:

printf("element which is deleted from queue is : %d\n\n",dequeue());

break;

case 3:

printf("Highest priority number in queue is : %d\n\n",highestpr());

break;

case 4:

display();

break;

case 5:

exit(1);

}

}

return 0;

}

void enqueue(int data,int p)

{

s[++top].item = data;

s[top].pr = p;

}

int dequeue()

{

if(top == -1){

printf("Queue is empty\n\n");

return -1000;}

else

{

int high = 10000;

int i,j;

for(i=0;i<=top;i++)

if(s[i].pr < high)

{

j=i;

high = s[i].pr;

}

int tmp = s[j].item;

for(i=j;i<top;i++)

{

s[i].item = s[i+1].item;

s[i].pr = s[i+1].pr;

}

top--;

return tmp;

}

}

int highestpr()

{

if(top == -1){

printf("Queue is empty\n");

return -1000;

}

else{

int i,j;

int high =10000;

for(i=0;i<=top;i++)

if(s[i].pr < high)

{

j = i;

high = s[i].pr;

}

return s[j].item;

}

}

void display()

{

if(top == -1)

printf("Queue is empty\n");

else

{

int arr[SIZE];

int i,j,min,tmp;

for(i=0;i<=top;i++)

arr[i] = s[i].pr;

for(i=0;i<top;i++){

for(j=i+1;j<=top;j++)

{

if(arr[i] > arr[j])

{

tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

}}}

printf("Queue in Priority order:\n");

for(i=0;i<=top;i++)

{

for(j=0;j<=top;j++)

if(s[j].pr == arr[i])

{

printf("%d\t",s[j].item);

break;

}

}

}

printf("\n\n");

}

***OUTPUT:***

