***ASSIGNMENT-4***

1. ***WAP to implement a stack using queues.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 20

struct queue{

int arr[SIZE];

int front,rear;

}q,q1,q2;

void push(int n,int data);

int pop();

int dequeue(int n);

void display();

int main()

{

q1.front = q2.front = 0;

q1.rear = q2.rear = -1;

int data,ch;

printf("Enter your choice\n");

printf("1.Push an element in stack\n");

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

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

printf("4.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("enter element you want to insert: ");

scanf("%d",&data);

push(1,data);

break;

case 2:

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

break;

case 3:

display();

break;

case 4:

exit(1);

}

q2.front = 0;

q2.rear = -1;

}

return 0;

}

void push(int n,int data)

{

if(n==1)

q1.arr[++(q1.rear)] = data;

else

q2.arr[++(q2.rear)] = data;

}

int pop()

{

if(q1.rear == -1)

{

printf("Stack underflow\n");

return -1000;

}

else

{

while(q1.front!=q1.rear)

push(2,dequeue(1));

int tmp = dequeue(1);

q = q1;

q1 = q2;

q2 = q1;

return tmp;

}

}

int dequeue(int n)

{

if(n==1)

return q1.arr[(q1.front)++];

else

return q2.arr[(q2.front)++];

}

void display()

{

if(q1.rear < q1.front)

printf("Stack is empty\n");

else

{

int i;

for(i=q1.front;i<=q1.rear;i++)

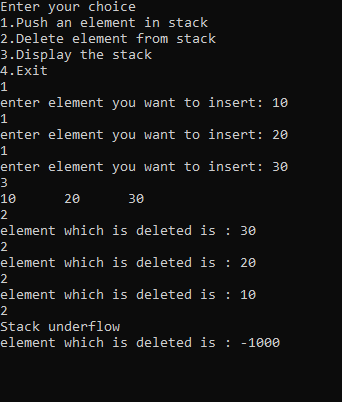
printf("%d\t",q1.arr[i]);

printf("\n");

}

}

***OUTPUT:***



1. ***WAP to implement a queue using stacks.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 20

struct stack{

int arr[SIZE];

int top;

}s1,s2;

void enqueue(int n,int data);

int dequeue();

int pop(int n);

void display();

int main()

{

s1.top = s2.top = -1;

int data,ch;

printf("Enter your choice\n");

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

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

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

printf("4.Exit\n");

while(1)

{

scanf("%d",&ch);

switch(ch)

{

case 1:

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

scanf("%d",&data);

enqueue(1,data);

break;

case 2:

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

break;

case 3:

display();

break;

case 4:

exit(1);

}

}

return 0;

}

void enqueue(int n,int data)

{

if(n==1)

s1.arr[++(s1.top)] = data;

if(n==2)

s2.arr[++(s2.top)] = data;

}

int dequeue()

{

if(s1.top == -1){

printf("Queue is empty\n");

return -1000;

}

while(s1.top>=0)

enqueue(2,pop(1));

int tmp = pop(2);

while(s2.top>=0)

enqueue(1,pop(2));

return tmp;

}

int pop(int n)

{

if(n==1)

return s1.arr[(s1.top)--];

else

return s2.arr[(s2.top)--];

}

void display()

{

if(s1.top<0)

printf("Queue is empty\n");

else

{

int i;

for(i=0;i<=s1.top;i++)

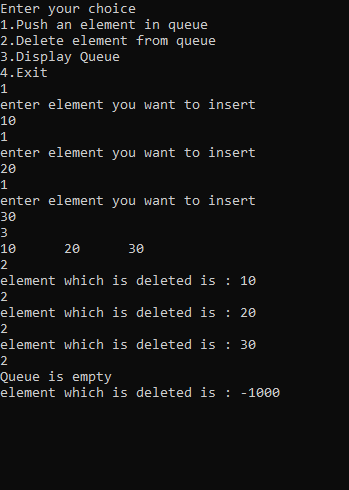
printf("%d\t",s1.arr[i]);

printf("\n");

}

}

***OUTPUT:***



1. ***WAP to implement n stacks in a single array.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

int n;

int base[SIZE];

struct stack{

int arr[SIZE];

int top[SIZE];

}s;

void push(int x,int data);

int pop(int x);

void display(int x);

int main()

{

int ch,x,data;

printf("Enter number of stacks\n");

scanf("%d",&n);

int i;

int l = SIZE/n;

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

base[i] = l\*i;

s.top[i] = l\*i -1;

}

base[i] = SIZE;

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

printf("1.Insertion\n");

printf("2.Deletion\n");

printf("3.Display\n");

printf("4.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter stack number and data to inserted: ");

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

push(x-1,data);

break;

case 2:

printf("Enter stack number: ");

scanf("%d",&x);

printf("Element which is deleted is : %d\n",pop(x-1));

break;

case 3:

printf("Enter stack number: ");

scanf("%d",&x);

display(x-1);

printf("\n");

break;

case 4:

exit(1);

}

}

return 0;

}

void push(int x,int data)

{

if((base[x+1]-s.top[x])>1)

s.arr[++(s.top[x])] = data;

else if(base[x+1]!=SIZE && ((base[x+2]-s.top[x+1])>1))

{

int i;

for(i=s.top[x+1]+1;i>base[x+1];i--)

s.arr[i] = s.arr[i-1];

s.arr[++(s.top[x])] = data;

base[x+1]++;

s.top[x+1]++;

}

else if(base[x]!=0 && ((base[x]-s.top[x-1])>1))

{

int i;

for(i=base[x]-1;i<s.top[x];i++)

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

s.arr[i]=data;

base[x]--;

}

else

printf("Stack overflow\n");

}

int pop(int x)

{

if(base[x]>s.top[x])

{

printf("Stack underflow\n");

return -100000;

}

else

return s.arr[(s.top[x])--];

}

void display(int x)

{

int i;

if(base[x]>s.top[x])

printf("Stack is empty\n");

else

{

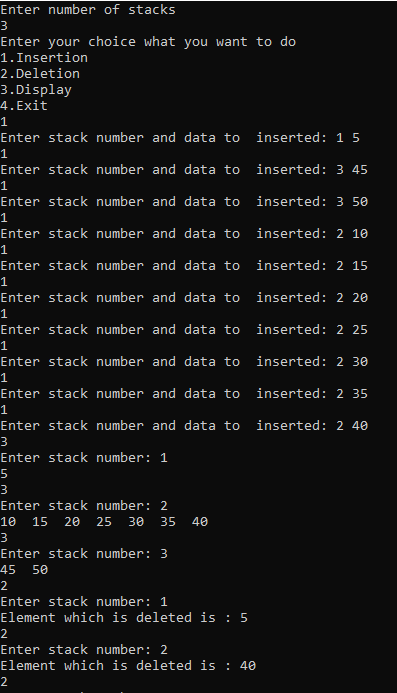
for(i=base[x];i<=s.top[x];i++)

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

}

}

***OUTOUT:***



1. ***WAP to implement n queues in a single array.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

int n;

struct queue{

int arr[SIZE];

int top[SIZE];

int front[SIZE];

}q;

void enqueue(int x,int data);

int dequeue(int x);

void display(int x);

int main()

{

int ch,x,data;

printf("Enter number of queue\n");

scanf("%d",&n);

int i;

int l = SIZE/n;

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

q.front[i] = l\*i;

q.top[i] = l\*i -1;

}

q.front[i] = SIZE;

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

printf("1.Insertion\n");

printf("2.Deletion\n");

printf("3.Display\n");

printf("4.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter queue number and data to inserted: ");

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

enqueue(x-1,data);

break;

case 2:

printf("Enter queue number: ");

scanf("%d",&x);

printf("Element which is deleted is : %d\n",dequeue(x-1));

break;

case 3:

printf("Enter queue number: ");

scanf("%d",&x);

display(x-1);

printf("\n");

break;

case 4:

exit(1);

}

}

return 0;

}

void enqueue(int x,int data)

{

if((q.front[x+1]-q.top[x])>1)

q.arr[++(q.top[x])] = data;

else if(q.front[x+1]!=SIZE && ((q.front[x+2]-q.top[x+1])>1))

{

int i;

for(i=q.top[x+1]+1;i>q.front[x+1];i--)

q.arr[i] = q.arr[i-1];

q.arr[++(q.top[x])] = data;

q.front[x+1]++;

q.top[x+1]++;

}

else if(q.front[x]!=0 && ((q.front[x]-q.top[x-1])>1))

{

int i;

for(i=q.front[x]-1;i<q.top[x];i++)

q.arr[i] = q.arr[i+1];

q.arr[i]=data;

q.front[x]--;

}

else

printf("Queue overflow\n");

}

int dequeue(int x)

{

if(q.front[x] > q.top[x])

{

printf("Queue underflow\n");

return -100000;

}

return q.arr[(q.front[x])++];

}

void display(int x)

{

if(q.front[x] > q.top[x])

printf("Queue is empty\n");

else

{

int i;

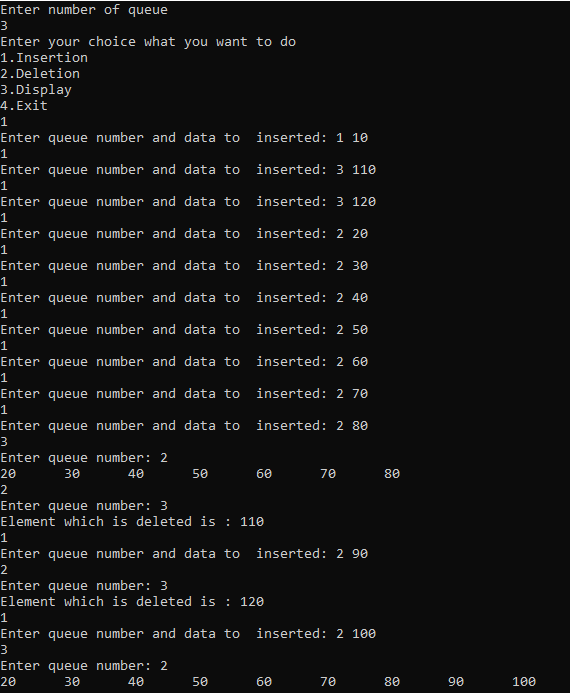
for(i=q.front[x];i<=q.top[x];i++)

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

}

}

***OUTOUT:***



***/\*end\*/***