**DSPLAB Assignment – 3**

1. Write a source code how to implement efficiently multiple queues in a single array

Source Code:

#include <stdio.h>

#define MAX 40

void insert\_que1(int \*,int \*,int \*,int);

void insert\_que2(int \*,int \*,int \*,int);

void insert\_que3(int \*,int \*,int \*,int);

void insert\_que4(int \*,int \*,int \*,int);

void delete\_que1(int \*,int \*,int \*,int \*);

void delete\_que2(int \*,int \*,int \*,int \*);

void delete\_que3(int \*,int \*,int \*,int \*);

void delete\_que4(int \*,int \*,int \*,int \*);

void display\_que1(int \*,int \*,int \*);

void display\_que2(int \*,int \*,int \*);

void display\_que3(int \*,int \*,int \*);

void display\_que4(int \*,int \*,int \*);

int c1=0;

int c2=0;

int c3=0;

int c4=0;

void main()

{

int que[MAX];

int front1,rear1,front2,rear2,front3,rear3,front4,rear4;

int value,choice,i;

front1 = rear1 = -1;

front2 = rear2 = 20;

front3 = rear3 = 20;

front4 = rear4 = 40;

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

que[i]=-1;

START:

do

{

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n 1. insert to queue1\n");

printf("\n 2. insert to queue2\n");

printf("\n 3. insert to queue3\n");

printf("\n 4. insert to queue4\n");

printf("\n 5. delete to queue1\n");

printf("\n 6. delete to queue2\n");

printf("\n 7. delete to queue3\n");

printf("\n 8. delete to queue4\n");

printf("\n 9. Display elements in Queue1\n");

printf("\n 10. Display elements in Queue2\n");

printf("\n 11. Display elements in Queue3\n");

printf("\n 12. Display elements in Queue4\n");

printf("\n 13. Display elements in ALLQueues\n");

printf("\n 14. Quit\n");

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n\tENTER CHOICE HERE :");

scanf("%d",&choice);

switch(choice)

{

case 1 : printf("Enter the element to be inserted:\n");

scanf("%d",&value);

insert\_que1(que,&front1,&rear1,value);

break;

case 2 : printf("Enter the element to be inserted:\n");

scanf("%d",&value);

insert\_que2(que,&front2,&rear2,value);

break;

case 3 : printf("Enter the element to be inserted:\n");

scanf("%d",&value);

insert\_que3(que,&front3,&rear3,value);

break;

case 4 : printf("Enter the element to be inserted:\n");

scanf("%d",&value);

insert\_que4(que,&front4,&rear4,value);

break;

case 5 : delete\_que1(que,&front1,&rear1,&value);

if(value == -1)

printf("Deque\_1 is empty\n");

else

printf("The value deleted from front is %d\n",value);

break;

case 6 : delete\_que2(que,&front2,&rear2,&value);

if(value == -1)

printf("Deque\_2 is empty\n");

else

printf("The value deleted from front is %d\n",value);

break;

case 7 : delete\_que3(que,&front3,&rear3,&value);

if(value == -1)

printf("Deque\_3 is empty\n");

else

printf("The value deleted from front is %d\n",value);

break;

case 8 : delete\_que4(que,&front4,&rear4,&value);

if(value == -1)

printf("Deque\_4 is empty\n");

else

printf("The value deleted from front is %d\n",value);

break;

case 9 : display\_que1(que,&front1,&rear1);

break;

case 10 : display\_que2(que,&front2,&rear2);

break;

case 11 : display\_que3(que,&front3,&rear3);

break;

case 12 : display\_que4(que,&front4,&rear4);

break;

case 13 : printf("\n\tMULTIPLE QUEUES:\n");

display\_que1(que,&front1,&rear1);

display\_que2(que,&front2,&rear2);

display\_que3(que,&front3,&rear3);

display\_que4(que,&front4,&rear4);

break;

case 14 : printf("\npress any key to QUIT !!!\n");

goto EXIT;

default: goto START;

}

}while(1);

EXIT:

getch();

}

void insert\_que1(int que[],int \*front1,int \*rear1,int value)

{

int i,k;

if(\*front1 == 0 && \*rear1 == MAX-11)

{

printf("Deque\_1 is full.\n");

return;

}

if(\*rear1 == -1)

{

\*rear1=\*front1=0;

que[\*rear1]=value;

c1++;

return;

}

if(\*rear1 == MAX-11)

{

printf("shifting");

k=\*front1-1;

for(i=1;i<=c1;i++)

{

que[k]=que[k+1];

k++;

}

que[k]=value;

\*rear1=k;

(\*front1)++;

c1++;

}

else

{

(\*rear1)++;

que[\*rear1]=value;

c1++;

}

}

void insert\_que2(int que[],int \*front2,int \*rear2,int value)

{

int i,k;

if(\*front2 == MAX-1 && \*rear2 == MAX-10)

{

printf("Deque\_2 is full.\n");

return;

}

if(\*rear2 == 20)

{

\*rear2=\*front2=19;

que[\*rear2]=value;

c2++;

return;

}

if(\*rear2 == MAX-10)

{

printf("shifting");

k=\*front2-1;

for(i=1;i<=c2;i++)

{

que[k]=que[k-1];

k--;

}

que[k]=value;

\*rear2=k;

(\*front2)--;

c2++;

}

else

{

(\*rear2)--;

que[\*rear2]=value;

c2++;

}

}

void insert\_que3(int que[],int \*front3,int \*rear3,int value)

{

int i,k;

if(\*front3 == 20 && \*rear3 == MAX-31)

{

printf("Deque\_3 is full.\n");

return;

}

if(\*rear3 == 20)

{

\*rear3=\*front3=21;

que[\*rear3]=value;

c3++;

return;

}

if(\*rear3 == MAX-31)

{

printf("shifting");

k=\*front3-21;

for(i=1;i<=c1;i++)

{

que[k]=que[k+1];

k++;

}

que[k]=value;

\*rear3=k;

(\*front3)++;

c3++;

}

else

{

(\*rear3)++;

que[\*rear3]=value;

c3++;

}

}

void insert\_que4(int que[],int \*front4,int \*rear4,int value)

{

int i,k;

if(\*front4 == MAX-21 && \*rear4 == MAX-30)

{

printf("Deque\_4 is full.\n");

return;

}

if(\*rear4 == 40)

{

\*rear4=\*front4=39;

que[\*rear4]=value;

c4++;

return;

}

if(\*rear4 == MAX-30)

{

printf("shifting");

k=\*front4-1;

for(i=1;i<=c4;i++)

{

que[k]=que[k-1];

k--;

}

que[k]=value;

\*rear4=k;

(\*front4)--;

c4++;

}

else

{

(\*rear4)--;

que[\*rear4]=value;

c4++;

}

}

void delete\_que1(int que[],int \*front1,int \*rear1,int \*value)

{

if(\*front1 == -1)

{

printf("Deque\_1 is empty\n");

\*value=-1;

return;

}

\*value = que[\*front1];

que[\*front1]=-1;

if(\*front1 == \*rear1)

{

\*front1=-1;

\*rear1=-1;

\*value=-1;

c1--;

}

else

{

(\*front1)++;

c1--;

}

}

void delete\_que2(int que[],int \*front2,int \*rear2,int \*value)

{

if(\*front2 == 20)

{

printf("Deque\_1 is empty\n");

\*value=-1;

return;

}

\*value = que[\*front2];

que[\*front2]=-1;

if(\*front2 == \*rear2)

{

\*front2=20;

\*rear2=20;

\*value=-1;

c2--;

}

else

{

(\*front2)--;

c2--;

}

}

void delete\_que3(int que[],int \*front3,int \*rear3,int \*value)

{

if(\*front3 == 20)

{

printf("Deque\_3 is empty\n");

\*value=20;

return;

}

\*value = que[\*front3];

que[\*front3]=-1;

if(\*front3 == \*rear3)

{

\*front3=20;

\*rear3=20;

\*value=20;

c3--;

}

else

{

(\*front3)++;

c3--;

}

}

void delete\_que4(int que[],int \*front4,int \*rear4,int \*value)

{

if(\*front4 == 40)

{

printf("Deque\_4 is empty\n");

\*value=20;

return;

}

\*value = que[\*front4];

que[\*front4]=20;

if(\*front4 == \*rear4)

{

\*front4=40;

\*rear4=40;

\*value=20;

c4--;

}

else

{

(\*front4)--;

c4--;

}

}

void display\_que1(int \*que,int \*front1,int\*rear1)

{

int i;

printf("\n QUEUE NO 1 :\t");

if(\*front1 == -1 || \*rear1 == -1)

printf("Deque\_1 is empty\n");

else

{ printf("front->");

for(i=\*front1;i<=\*rear1;i++)

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

printf(" <-rear");

}

}

void display\_que2(int \*que,int \*front2,int\*rear2)

{

int i;

printf("\n QUEUE NO 2 :\t");

if(\*front2 == 20 || \*rear2 == 20)

printf("Deque\_2 is empty\n");

else

{ printf("front->");

for(i=\*front2;i>=\*rear2;i--)

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

printf(" <-rear");

}

}

void display\_que3(int \*que,int \*front3,int\*rear3)

{

int i;

printf("\n QUEUE NO 3 :\t");

if(\*front3 == 20 || \*rear3 == 20)

printf("Deque\_3 is empty\n");

else

{ printf("front->");

for(i=\*front3;i<=\*rear3;i++)

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

printf(" <-rear");

}

}

void display\_que4(int \*que,int \*front4,int\*rear4)

{

int i;

printf("\n QUEUE NO 4 :\t");

if(\*front4 == 40 || \*rear4 == 40)

printf("Deque\_4 is empty\n");

else

{ printf("front->");

for(i=\*front4;i>=\*rear4;i--)

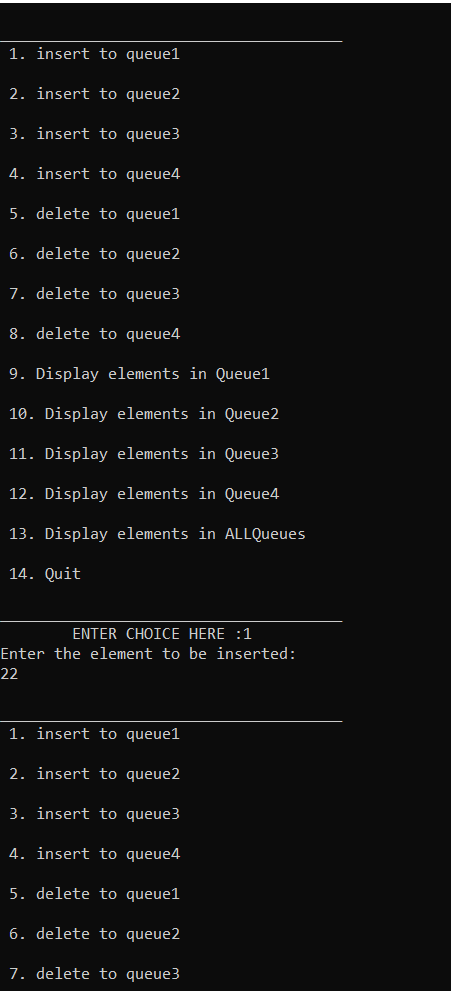
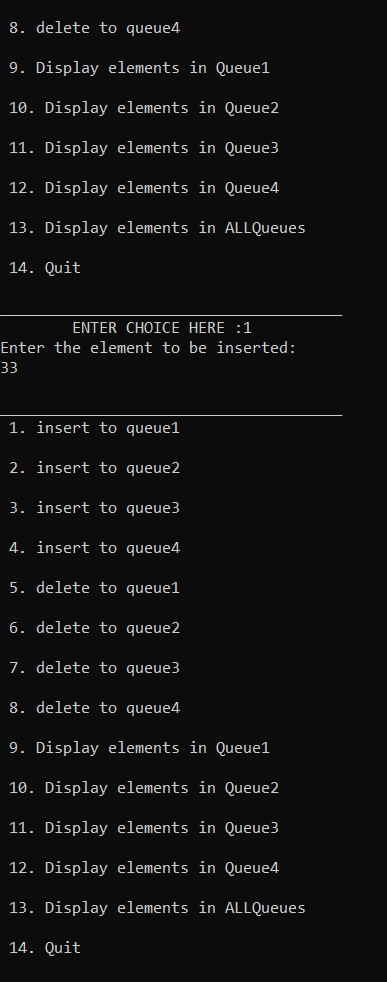
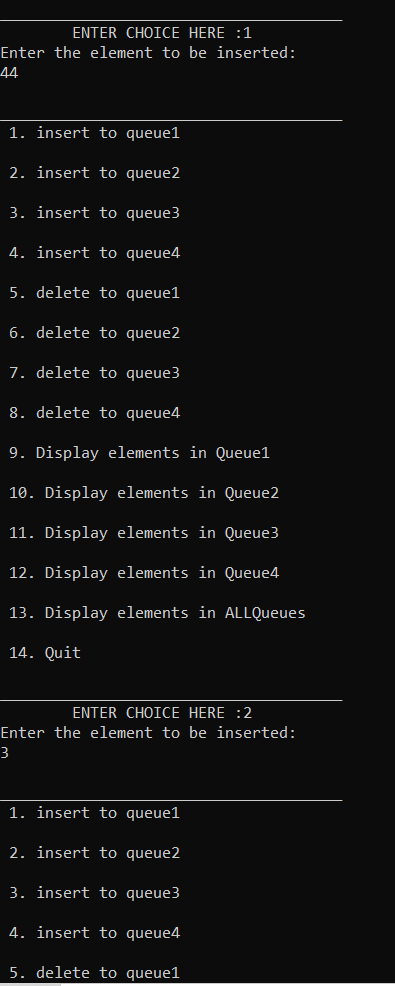
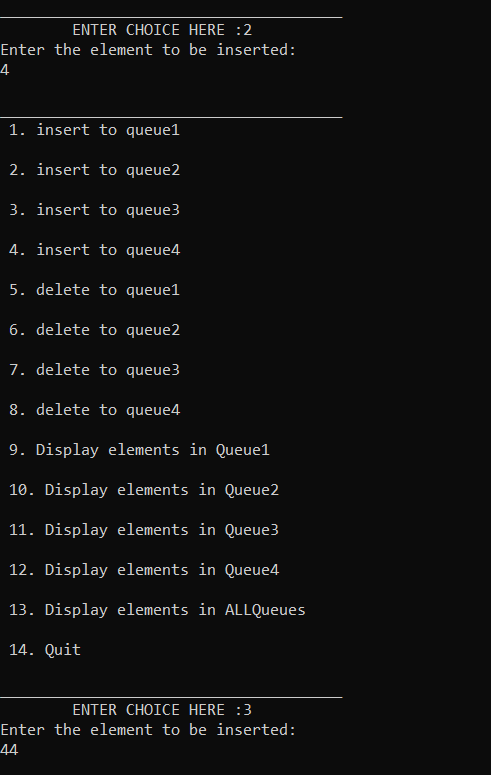
printf(" %d",que[i]);

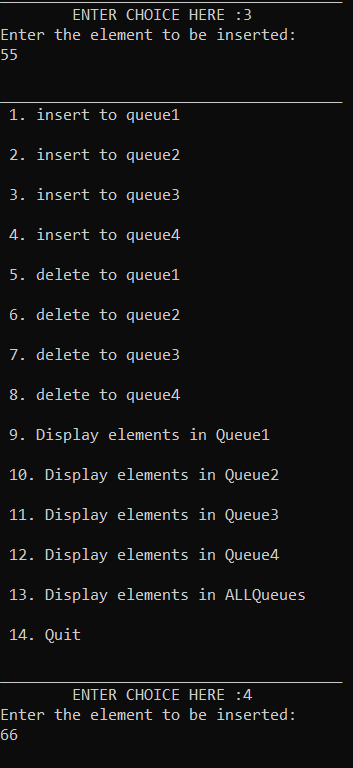
printf(" <-rear");

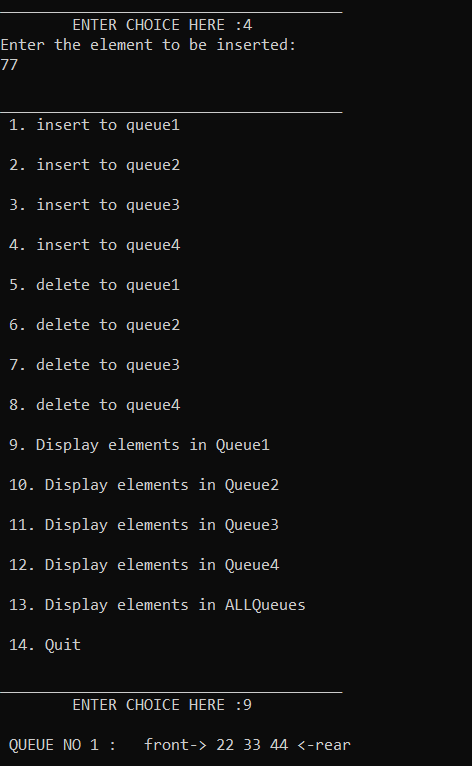
}

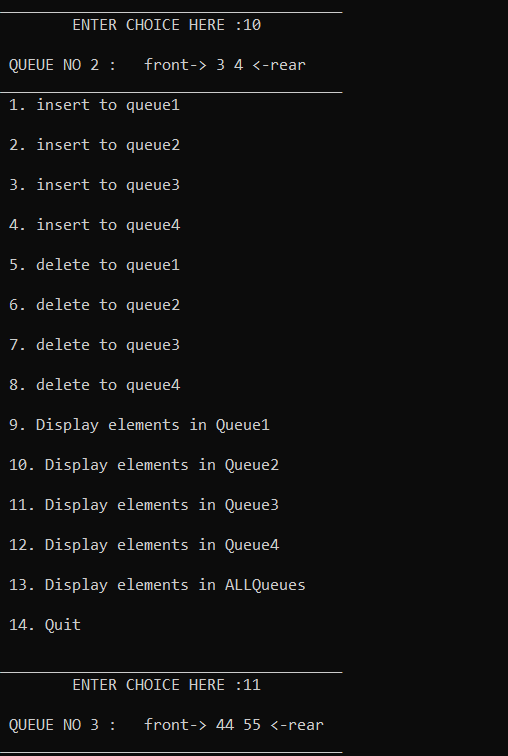
}

Output:







Output2:

