NAME: KUSUM M R **DATE: 09/12/2020**
USN:1BM19CS077

Lab 8:CODE and OUTPUT
WAP to implement Stack & Queues using Linked Representation

CODE:

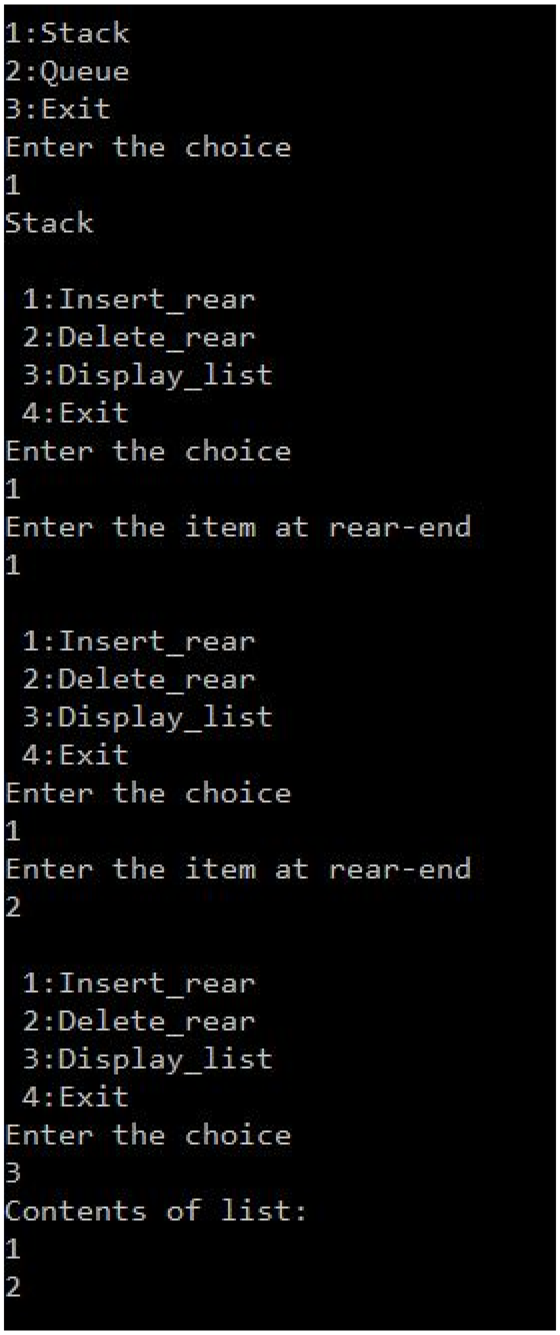
```
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("Memory full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
```

```
return first;
}

NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}


NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("List is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("Item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}


void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 {
 printf("List empty cannot display items\n");
 return;
 }
 printf("Contents of list:\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}

void main()
{
int item,choice,pos,i,n,count,key;
NODE first=NULL,a,b;
```

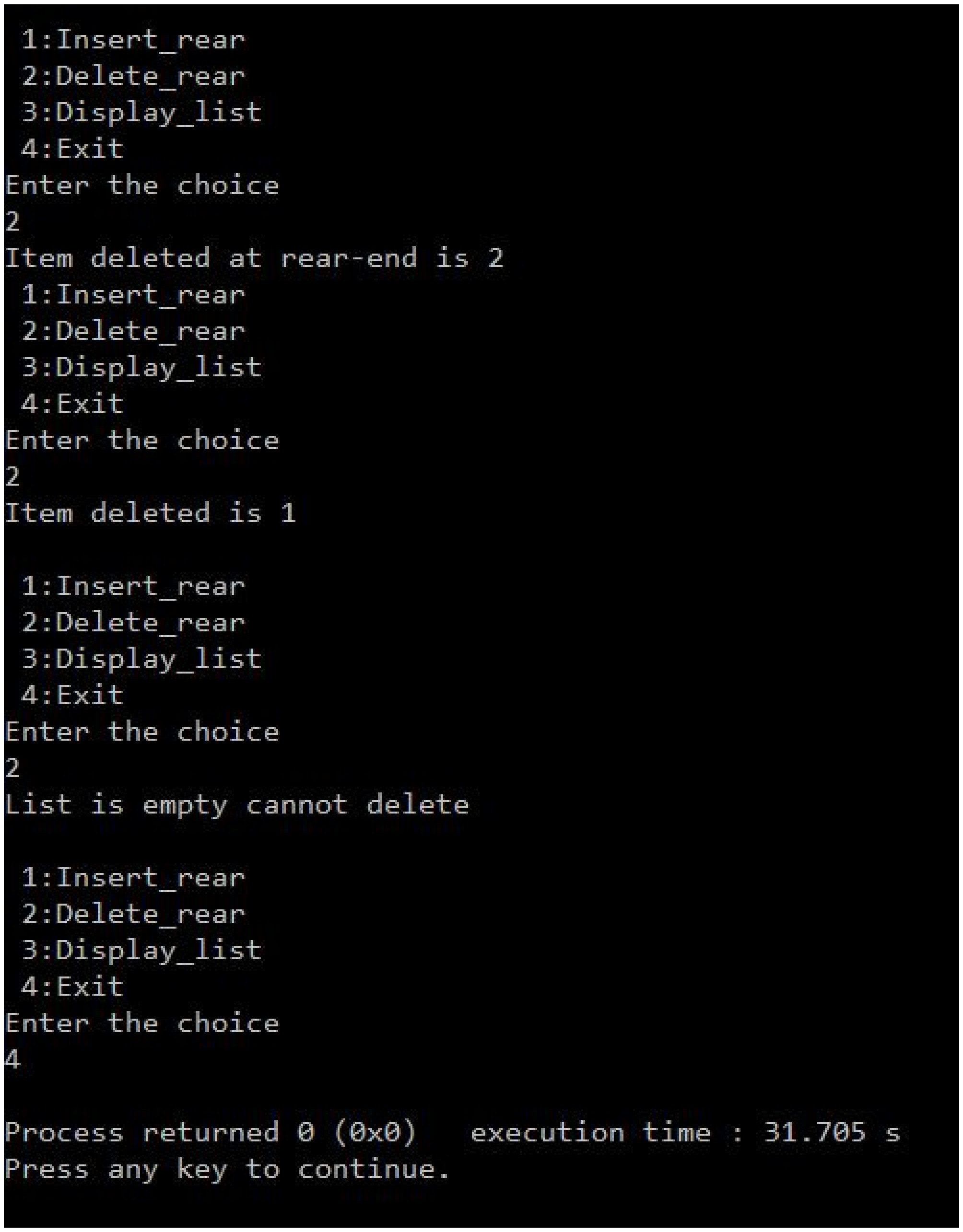
```
for(;;)
{
printf("\n1:Stack\n2:Queue\n3:Exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {

  case 1:printf("Stack\n");
      for(;;)
      {
        printf("\n 1:Insert_rear\n 2:Delete_rear\n 3:Display_list\n 4:Exit\n");
        printf("Enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:printf("Enter the item at rear-end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
        case 2:first=delete_rear(first);
            break;
        case 3:display(first);
            break;
        default:exit(0);
            break;
        }
      }
  case 2:printf("QUEUE\n");
        for(;;)
        {
            printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
            printf("Enter the choice\n");
            scanf("%d",&choice);
            switch(choice)
            {
            case 1:printf("Enter the item at rear-end\n");
                    scanf("%d",&item);
                    first=insert_rear(first,item);
```

```
                break;
            case 2:first=delete_front(first);
                break;
            case 3:display(first);
                break;
            default:exit(0);
                break;
            }
          }

  case 3:exit(0);
   default:printf("Invalid choice\n");
  }
 }
 }
```

OUTPUT:
CASE 1:Stack

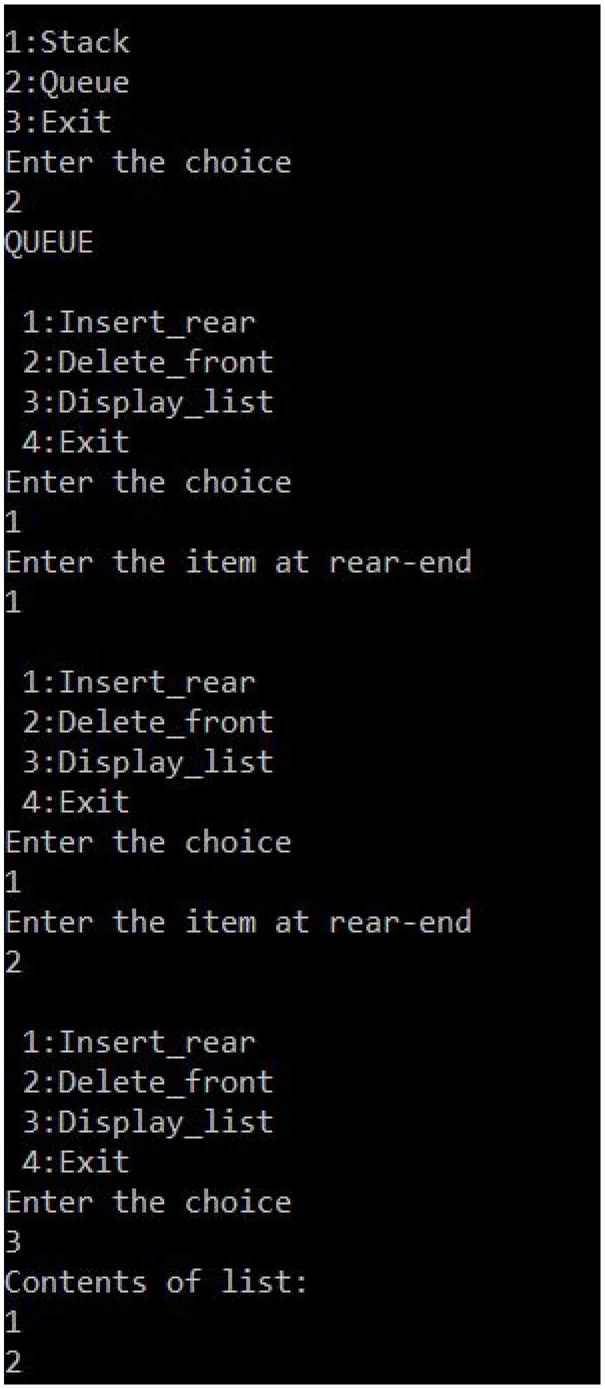


```
1:Stack
2:Queue
3:Exit
Enter the choice
1
Stack

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
1

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
2

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
1
2
```

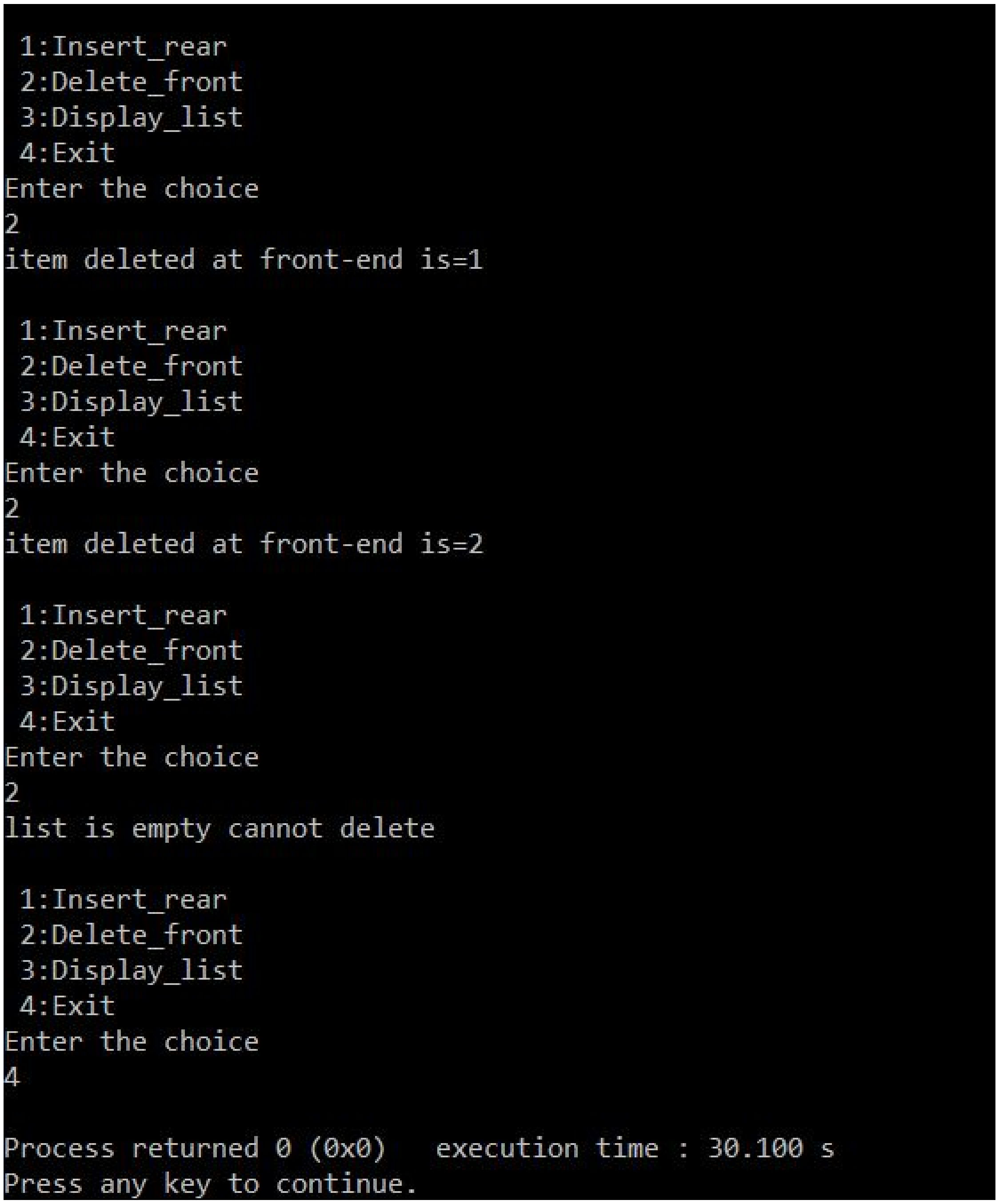


```
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 2
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted is 1

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
List is empty cannot delete

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
4

Process returned 0 (0x0)   execution time : 31.705 s
Press any key to continue.
```

CASE 2:Queue

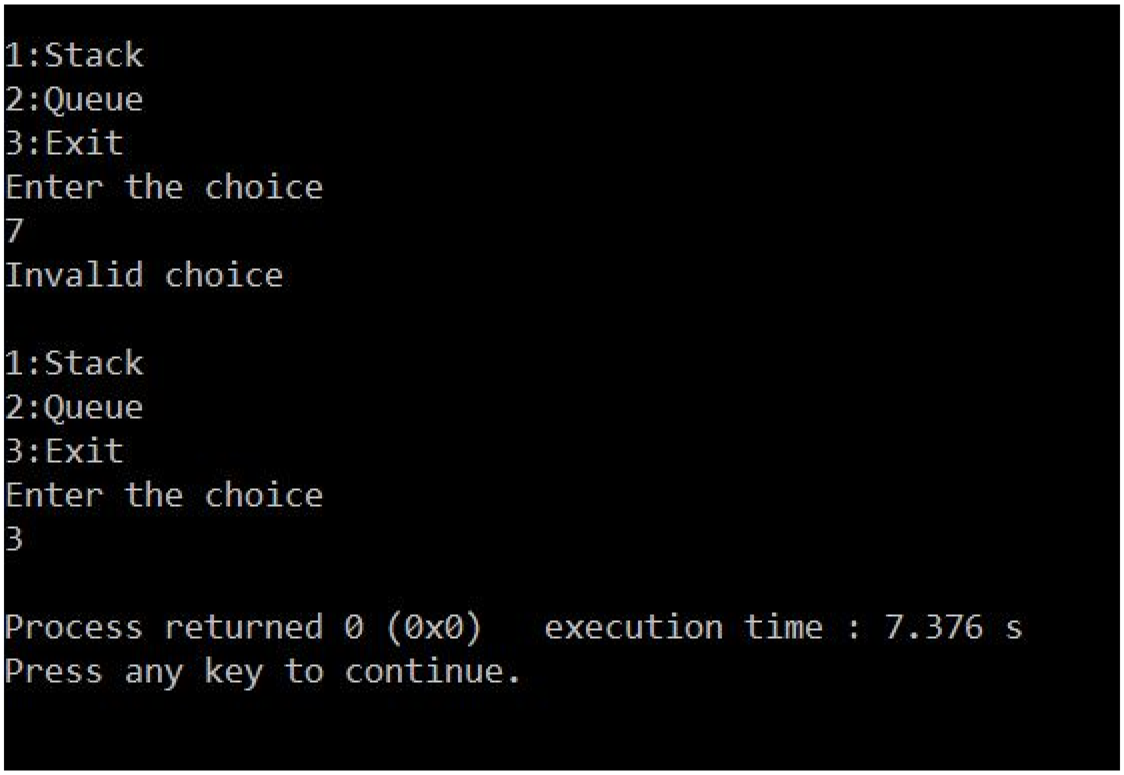


```
1:Stack
2:Queue
3:Exit
Enter the choice
2
QUEUE

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
1

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
2

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
1
2
```



```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=1

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=2

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
list is empty cannot delete

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
4

Process returned 0 (0x0)   execution time : 30.100 s
Press any key to continue.
```

Case 3: Invalid choice and Exit



```
1:Stack
2:Queue
3:Exit
Enter the choice
7
Invalid choice

1:Stack
2:Queue
3:Exit
Enter the choice
3

Process returned 0 (0x0)   execution time : 7.376 s
Press any key to continue.
```