Set A

a) Implement a linear queue library (st_queue.h) of integers using a static implementation of the queue and implementing the above six operations. Write a program that includes queue library and calls different queue operations

```
#include<stdio.h>
#include"stqueue.h"

main()
{
int n,i=0,ch;
struct node p;
init(&p);
do
{
printf("\nchoices are:\n1:push/insert into queue\n2:pop/delete from queue\n3:check whether queue is empty or not\n4:check whether queue is full or not\n5:peek(top) element of queue\n6:exit\n");
printf("enter your choice: ");
scanf("%d",&ch);

switch(ch)
{
case 1:printf("\nenter element:");
      scanf("%d",&n);
      add(&p,n);
      break;

case 2:printf("deleted elements is:%d",del(&p));
      break;

case 3:i=isempty(&p);
      if(i==1)
      printf("\nqueue is empty");
      else
      printf("\nqueue is not empty");
      break;
```

```
case 4:i=isfull(&p);
       if(i==1)
       printf("\nqueue is full");
       else
       printf("queue is not full");
       break;

case 5://n1=peek();
       printf("top element of queue is:%d",peek(&p));
       break;

case 6:exit(0);
}
}  while(ch!=6);
}
```

//st_queue.h

```
struct node
{
   int data;
   struct node *link;
}*front, *rear;
void insert();
void delete();
void queue_size();
void check();
void first_element();
void insert()
{
   struct node *temp;

   temp = (struct node*)malloc(sizeof(struct node));
   printf("Enter value to be inserted \n");
   scanf("%d", &temp->data);
   temp->link = NULL;
   if (rear  ==  NULL)
   {
      front = rear = temp;
```

```
    }
    else
    {
        rear->link = temp;
        rear = temp;
    }
}
void delete()
{
    struct node *temp;

    temp = front;
    if (front == NULL)
    {
        printf("queue is empty \n");
        front = rear = NULL;
    }
    else
    {
        printf("deleted element is %d\n", front->data);
        front = front->link;
        free(temp);
    }
}
void check()
{
    if (front == NULL)
        printf("\nQueue is empty\n");
    else
        printf("*************** Elements are present in the queue
**************\n");
}
void first_element()
{
    if (front == NULL)
    {
        printf("**************** The queue is empty ***************\n");
    }
    else
```

```
        printf("***************** The front element is %d ***********\n",
front->data);
}
void queue_size()
{
   struct node *temp;

   temp = front;
   int cnt = 0;
   if (front  ==  NULL)
   {
      printf(" queue empty \n");
   }
   while (temp)
   {
      printf("%d  ", temp->data);
      temp = temp->link;
      cnt++;
   }
   printf("********* size of queue is %d ******** \n", cnt);
}
```

Set B

a) Implement a linear queue library (dyqueue.h) of integers using a dynamic (circular linked list) implementation of the queue and implementing the above five operations. Write a driver program that includes queue library and calls different queue operations.

```
#include <stdio.h>
#include <stdlib.h>
#include "dyqueue.h"
#define MAX 10
void main()
{
   int choice, value;

   while(1)
   {
      printf("enter the choice \n");
```

```
        printf("1 : display size of queue \n2 : Insert element\n");
        printf("3 : Dequeue an element \n4 : Check if empty\n");
        printf("5. Get the first element of the queue\n");
        printf("6. Get the number of entries in the queue\n");
        printf("7. Exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
           printf("queue is created with a capacity of %d\n", MAX);
           break;
        case 2:
           insert();
           break;
        case 3:
           delete();
           break;
        case 4:
           check();
           break;
        case 5:
           first_element();
           break;
        case 6:
           queue_size();
           break;
        case 7:
           exit(0);
        default:
           printf("wrong choice\n");
           break;
        }
    }
}

//dyqueue.h

struct node
{
    int data;
```

```
    struct node *link;
}*front, *rear;
void insert();
void delete();
void queue_size();
void check();
void first_element();
void insert()
{
    struct node *temp;

    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter value to be inserted \n");
    scanf("%d", &temp->data);
    temp->link = NULL;
    if (rear  ==  NULL)
    {
        front = rear = temp;
    }
    else
    {
        rear->link = temp;
        rear = temp;
    }
}
void delete()
{
    struct node *temp;

    temp = front;
    if (front == NULL)
    {
        printf("queue is empty \n");
        front = rear = NULL;
    }
    else
    {
        printf("deleted element is %d\n", front->data);
        front = front->link;
        free(temp);
```

```
    }
}
void check()
{
    if (front == NULL)
        printf("\nQueue is empty\n");
    else
        printf("*************** Elements are present in the queue
**************\n");
}
void first_element()
{
    if (front == NULL)
    {
        printf("**************** The queue is empty ***************\n");
    }
    else
        printf("**************** The front element is %d **********\n",
front->data);
}
void queue_size()
{
    struct node *temp;

    temp = front;
    int cnt = 0;
    if (front  ==  NULL)
    {
        printf(" queue empty \n");
    }
    while (temp)
    {
        printf("%d  ", temp->data);
        temp = temp->link;
        cnt++;
    }
    printf("********* size of queue is %d ******** \n", cnt);
}
```