DS LAB

Lab 6

Q1)

Source Code:
"ascpriq.h"

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5


typedef struct
{
   int items[MAX];
   int front, rear;
} PQUEUE;

void pqinsert(PQUEUE * pq, int);
void pqmindelete(PQUEUE * pq);
void create(PQUEUE * pq);
void ins(PQUEUE * pq, int);
void pqdisplay(PQUEUE * pq);

void create(PQUEUE * pq)
{
   pq->front = -1;
   pq->rear = -1;
}

void pqinsert(PQUEUE * pq, int x)
{
   if (pq->rear >= MAX - 1)
   {
      printf("Queue Overflow\n");
      return;
   }
   if ((pq->front == -1) && (pq->rear == -1))
   {
      pq->front++;
      pq->rear++;
      pq->items[pq->rear] = x;
      return;
   }
   else
      ins(pq, x);
   pq->rear++;
}
void ins(PQUEUE * pq, int x)
{
```

```
    int i, j;
    for (i = 0; i <= pq->rear; i++)
    {
        if (x <= pq->items[i])
        {
            for (j = pq->rear + 1; j > i; j--)
            {
                pq->items[j] = pq->items[j - 1];
            }
            pq->items[i] = x;
            return;
        }
    }
    pq->items[i] = x;
}

void pqmindelete(PQUEUE * pq)
{
    int i;
    if ((pq->front == -1) && (pq->rear == -1))
    {
        printf("Queue Empty\n");
        return;
    }
    for (i = 0; i < pq->rear; i++)
    {
        pq->items[i] = pq->items[i + 1];
    }
    pq->items[i] = -99;
    pq->rear--;
    if (pq->rear == -1)
        pq->front = -1;
    return;
}
void pqdisplay(PQUEUE * pq)
{
    if ((pq->front == -1) && (pq->rear == -1))
    {
        printf("\nQueue is empty");
        return;
    }
    for (; pq->front <= pq->rear; pq->front++)
    {
        printf(" %d ", pq->items[pq->front]);
    }
    pq->front = 0;
}
```

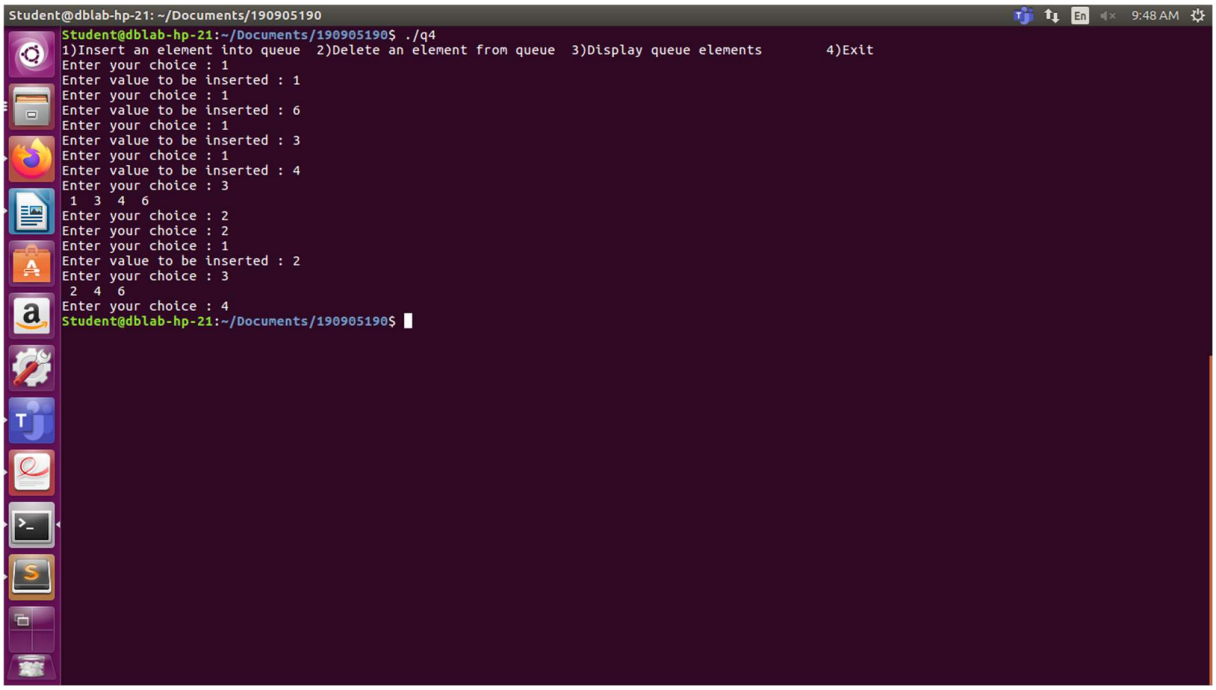
“q4.c”

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
```

```
#include "ascpriq.h"

void main()
{
   PQUEUE * pq;
   pq = malloc(sizeof(PQUEUE));
   int n, ch;
   printf("1)Insert an element into queue");
   printf("\t2)Delete an element from queue");
   printf("\t3)Display queue elements");
   printf("\t4)Exit\n");
   create(pq);
   while (1)
   {
      printf("Enter your choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         printf("Enter value to be inserted : ");
         scanf("%d", &n);
         pqinsert(pq, n);
         break;
      case 2:
         pqmindelete(pq, n);
         break;
      case 3:
         pqdisplay(pq);
         printf("\n");
         break;
      case 4:
         exit(0);
      default:
         printf("\nChoice is incorrect, Enter a correct choice");
      }
   }
}
```

Output:



Q2)

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXSIZE 5
#define MAXLENGTH 100

typedef struct
{
	char item[MAXSIZE][MAXLENGTH];
	int front;
	int rear;
} DQUEUE;

void init(DQUEUE * dq);
int isEmpty(DQUEUE * dq);
int isFull(DQUEUE * dq);
void insertLeft(DQUEUE * dq, char ele[]);
void insertRight(DQUEUE * dq, char ele[]);
char * deleteLeft(DQUEUE * dq);
void display(DQUEUE * dq);

void init(DQUEUE * dq)
{
	dq->front = -1;
	dq->front = -1;
}
```

```
int isEmpty(DQUEUE * dq)
{
	if(dq->rear == -1)
	{
		return 1;
	}
	else
		return 0;
}

int isFull(DQUEUE * dq)
{
	if((dq->rear + 1)%MAXSIZE == dq->front)
		return 1;
	else
		return 0;
}

void insertLeft(DQUEUE * dq, char ele[])
{
	if(isEmpty(dq))
	{
		dq->rear = dq->front = 0;
		strcpy(dq->item[dq->front], ele);
	}
	else
	{
		dq->front = (dq->front - 1 + MAXSIZE) % MAXSIZE;
		strcpy(dq->item[dq->front], ele);
	}
}

void insertRight(DQUEUE * dq, char ele[])
{
	if(isEmpty(dq))
	{
		dq->rear = dq->front = 0;
		strcpy(dq->item[dq->rear], ele);
	}
	else
	{
		dq->rear = (dq->rear + 1) % MAXSIZE;
		strcpy(dq->item[dq->rear], ele);
	}
}

char * deleteLeft(DQUEUE * dq)
{
	char * str;
	str = dq->item[dq->front];
	if(dq->rear == dq->front)
	{
```

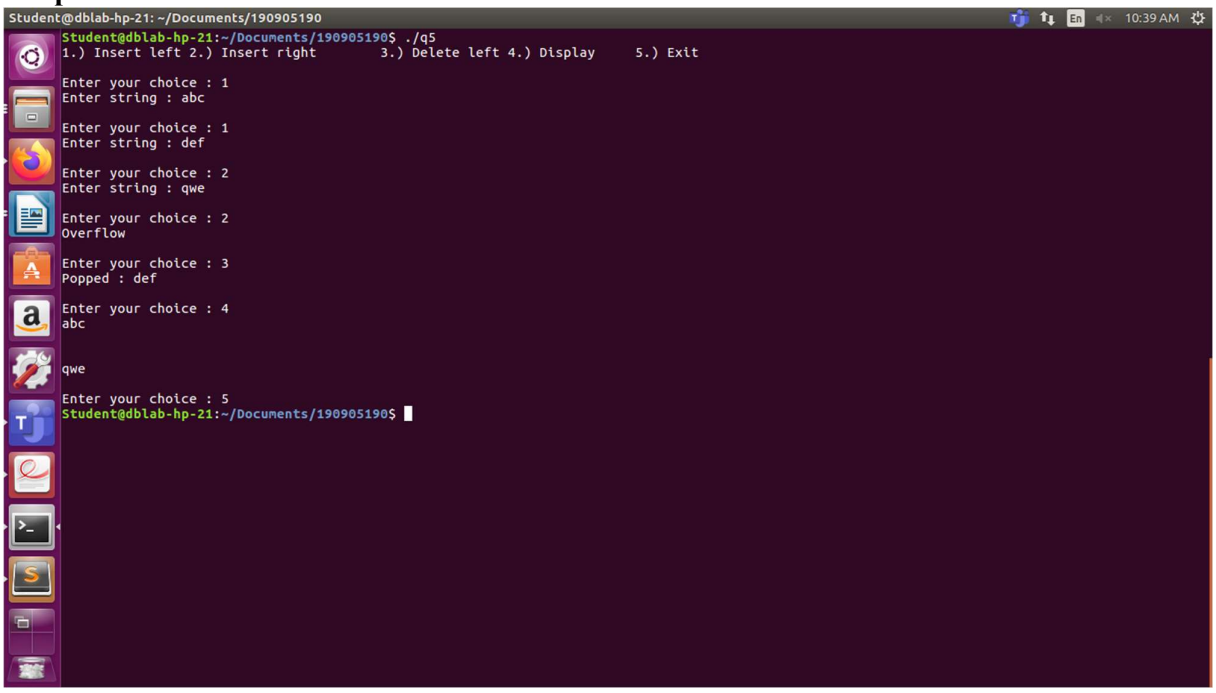
```
                init(dq);
        }
        else
        {
                dq->front = (dq->front + 1) % MAXSIZE;
        }
        return str;
}

void display(DQUEUE * dq)
{
        if(isEmpty(dq))
        {
                printf("Queue is Empty\n");
                return;
        }
        for (int temp = (dq->front) % MAXSIZE; temp != (dq->rear); temp = (temp + 1) %
MAXSIZE)
      printf("%s\n", dq->item[temp]);
   printf("%s\n", dq->item[dq->rear]);
}

int main()
{
   DQUEUE * dq;
   dq = malloc(sizeof(DQUEUE));
   init(dq);
   int ch;
   char str[MAXLENGTH];
   printf("1.) Insert left\t2.) Insert right\t3.) Delete left\t4.) Display\t5.) Exit\n");
   while (1)
   {
      printf("\nEnter your choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         if (isFull(dq))
            printf("Overflow\n");
         else
         {
            printf("Enter string : ");
            scanf("%s", str);
            insertLeft(dq, str);
         }
         break;
      case 2:
         if (isFull(dq))
            printf("Overflow\n");
         else
         {
            printf("Enter string : ");
```

```
            scanf(" %s", str);
            insertRight(dq, str);
        }
        break;
    case 3:
        if (!isEmpty(dq))
        {
            char * pop = deleteLeft(dq);
            printf("Popped : %s\n", pop);
        }
        else
            printf("Underflow\n");
        break;
    case 4:
        display(dq);
        break;
    case 5:
        exit(0);
    default:
        printf("Wrong Choice! Try Again");
    }
  }
}
```

Output:



Q3)

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 30

typedef struct DQUEUE
{
   char item[MAX];
   int rear, front;
} DQUEUE;

void init(DQUEUE *dq)
{
   dq->rear = -1;
   dq->front = -1;
}
int isEmpty(DQUEUE *dq)
{
   if (dq->rear == -1)
      return (1);
   return (0);
}
int full(DQUEUE *dq)
{
   if ((dq->rear + 1) % MAX == dq->front)
      return (1);
   return (0);
}
void insertRight(DQUEUE *dq, char x)
{
   if (isEmpty(dq))
   {
      dq->rear = 0;
      dq->front = 0;
      dq->item[0] = x;
   }
   else
   {
      dq->rear = (dq->rear + 1) % MAX;
      dq->item[dq->rear] = x;
   }
}
void insertLeft(DQUEUE *dq, char x)
{
   if (isEmpty(dq))
   {
      dq->rear = 0;
      dq->front = 0;
      dq->item[0] = x;
```

```
    }
    else
    {
        dq->front = (dq->front - 1 + MAX) % MAX;
        dq->item[dq->front] = x;
    }
}
char deleteFront(DQUEUE *dq)
{
    char x;
    x = dq->item[dq->front];
    if (dq->rear == dq->front)
        /*delete the last element */
        init(dq);
    else
        dq->front = (dq->front + 1) % MAX;
    return (x);
}
char deleteRight(DQUEUE *dq)
{
    char x;
    x = dq->item[dq->rear];
    if (dq->rear == dq->front)
        init(dq);
    else
        dq->rear = (dq->rear - 1 + MAX) % MAX;
    return (x);
}
void print(DQUEUE *dq)
{
    if (isEmpty(dq))
    {
        printf("\nQueue is empty!!");
        exit(0);
    }
    int i;
    i = dq->front;
    while (i != dq->rear)
    {
        printf("\n%c", dq->item[i]);
        i = (i + 1) % MAX;
    }
    printf("\n%c\n", dq->item[dq->rear]);
}
int main()
{
    int i, x, n;
    int op = 0;
    char c[20];
    DQUEUE q;
    init(&q);
    printf("Enter string to check for palindrome\n");
```

```
    scanf("%s", c);
    n = strlen(c);
    for (i = 0; i < n; i++)
    {
        insertLeft(&q, c[i]);
    }
    for (i = 0; i < n / 2; i++)
    {
        if (deleteFront(&q) != deleteRight(&q))
        {
            op = 1;
            break;
        }
    }
    if (op == 0)
        printf("%s is palindrome\n", c);
    else
        printf("%s is not palindrome\n", c);
    return 0;
}
```

Output: