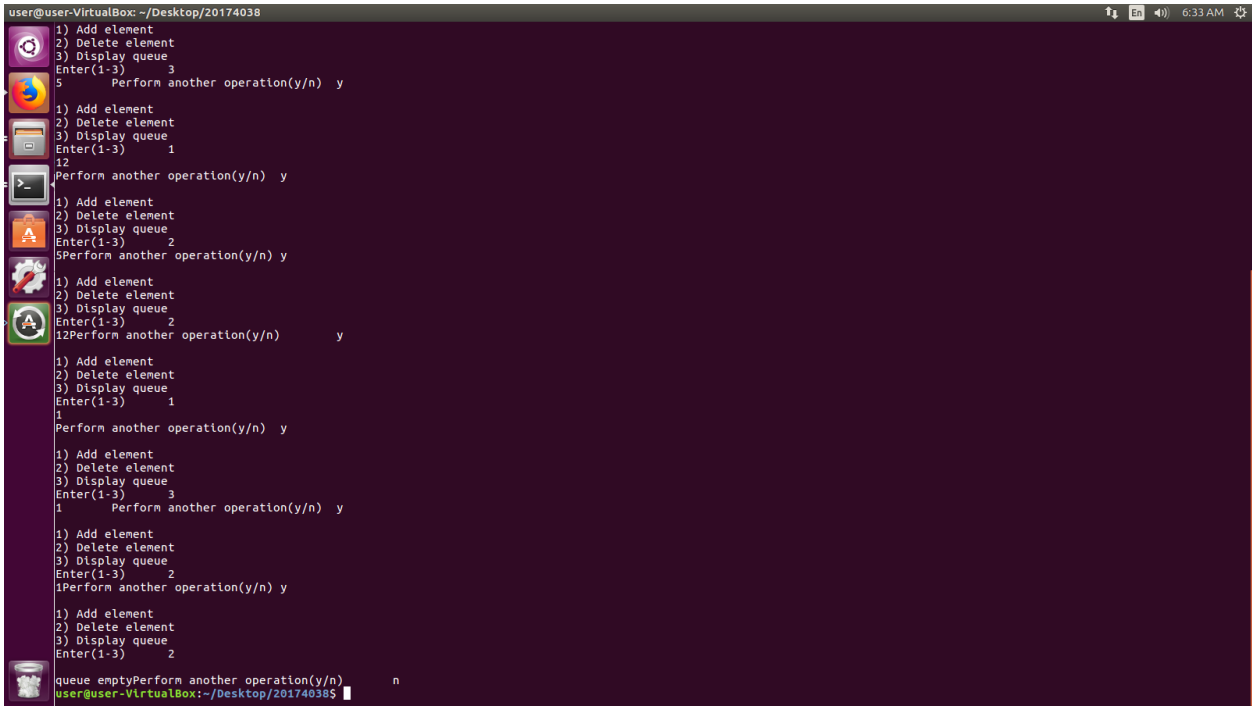
Q1)

```
#include<stdio.h>
#define mx 100
struct queue{
   int A[mx];
   int front;
   int rear;
};
typedef struct queue Q;
void enqueue(int y, Q * q) {
   int x;
   if(q->rear!=mx)
   q->A[q->rear++]=y;
   else if(q->front!=-1) {
          q->front--;
          x=q->front;
          while(x!=q->rear-1) {
                 q->A[x]=q->A[x+1];
          }
          q->A[q->rear-1]=y;
   }
   else
   printf("\nOverflow\n");
}
int dequeue(Q *q) {
if((q->rear)-(q->front)-1==0)
printf("\nqueue empty");
else {
printf("%d",q->A[++(q->front)]);
return (q->A[q->front]);
}
}
void display(Q *q){
int x=q->front+1;
while(x!=q->rear)
printf("%d\t",q->A[x++]);
}
void main(){
Q q;
int ch;
int x;
```

```
q.rear=0;
q.front=-1;
char t='y';
while(t=='y'||t=='Y'){
printf("\n1) Add element\n2) Delete element\n3) Display queue\nEnter(1-3)\t");
scanf("%d",&ch);
switch(ch) {
case 1: scanf("%d",&x);
   enqueue(x,&q);
   break;
case 2: dequeue(&q);
   break;
case 3: display(&q);
}
printf("Perform another operation(y/n)\t");
scanf("%c",&t);
scanf("%c",&t);
}
}
```

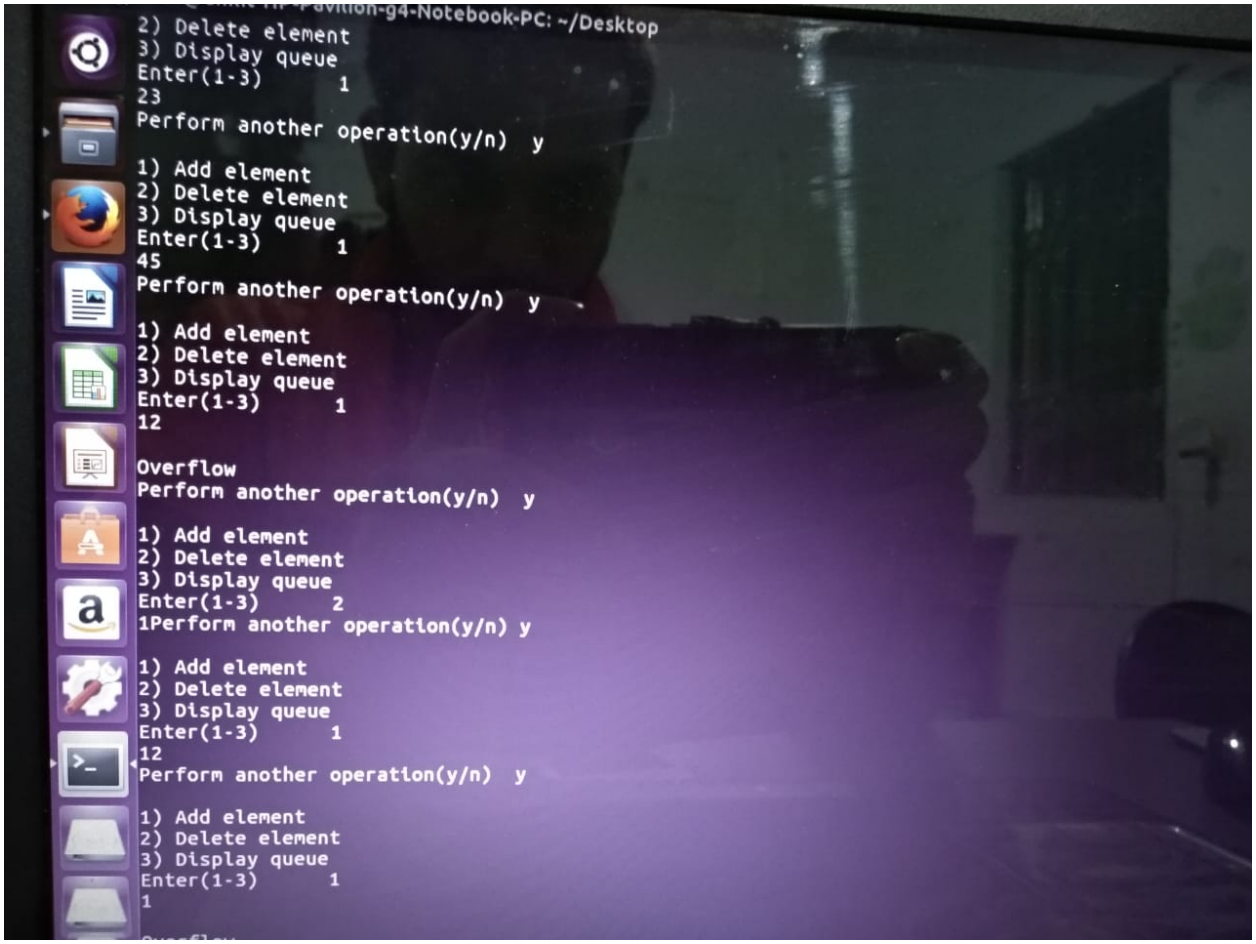


Q2)

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

```
#define mx 3
struct queue{
   int A[mx];
   int front;
   int rear;
   int ne;
};
typedef struct queue Q;
void enqueue(int y, Q * q) {
   int x;
   if(q->rear%mx!=(q->front+1)%mx)
   {
         q->A[(q->rear++)%mx]=y;
         q->ne++;
   }
   else if(q->ne==0) {
         q->A[(q->rear++)%mx]=y;
         q->ne++;
   }
   else
   printf("\nOverflow\n");
}
int dequeue(Q *q) {
   if(q->rear%mx!=(q->front+1)%mx)
   {
         printf("%d",q->A[(++(q->front))%mx]);
         q->ne--;
   }
   else if(q->ne==0)
   {
         printf("\nqueue empty");
         return;
   }
   else
   {
         printf("%d",q->A[(++(q->front))%mx]);
         q->ne--;
   }
   return (q->A[q->front%mx]);
}
void display(Q *q){
int x=(q->front+1)%mx;
int y=q->ne;
```

```
while(y--)
printf("%d\t",q->A[(x++)%mx]);
}
void main(){
Q q;
int ch;
int x;
q.rear=0;
q.front=mx-1;
q.ne=0;
char t='y';
while(t=='y'||t=='Y'){
printf("\n1) Add element\n2) Delete element\n3) Display queue\nEnter(1-3)\t");
scanf("%d",&ch);
switch(ch) {
case 1: scanf("%d",&x);
   enqueue(x,&q);
   break;
case 2: dequeue(&q);
   break;
case 3: display(&q);
}
printf("Perform another operation(y/n)\t");
scanf("%c",&t);
scanf("%c",&t);
}
}
```



Q3)

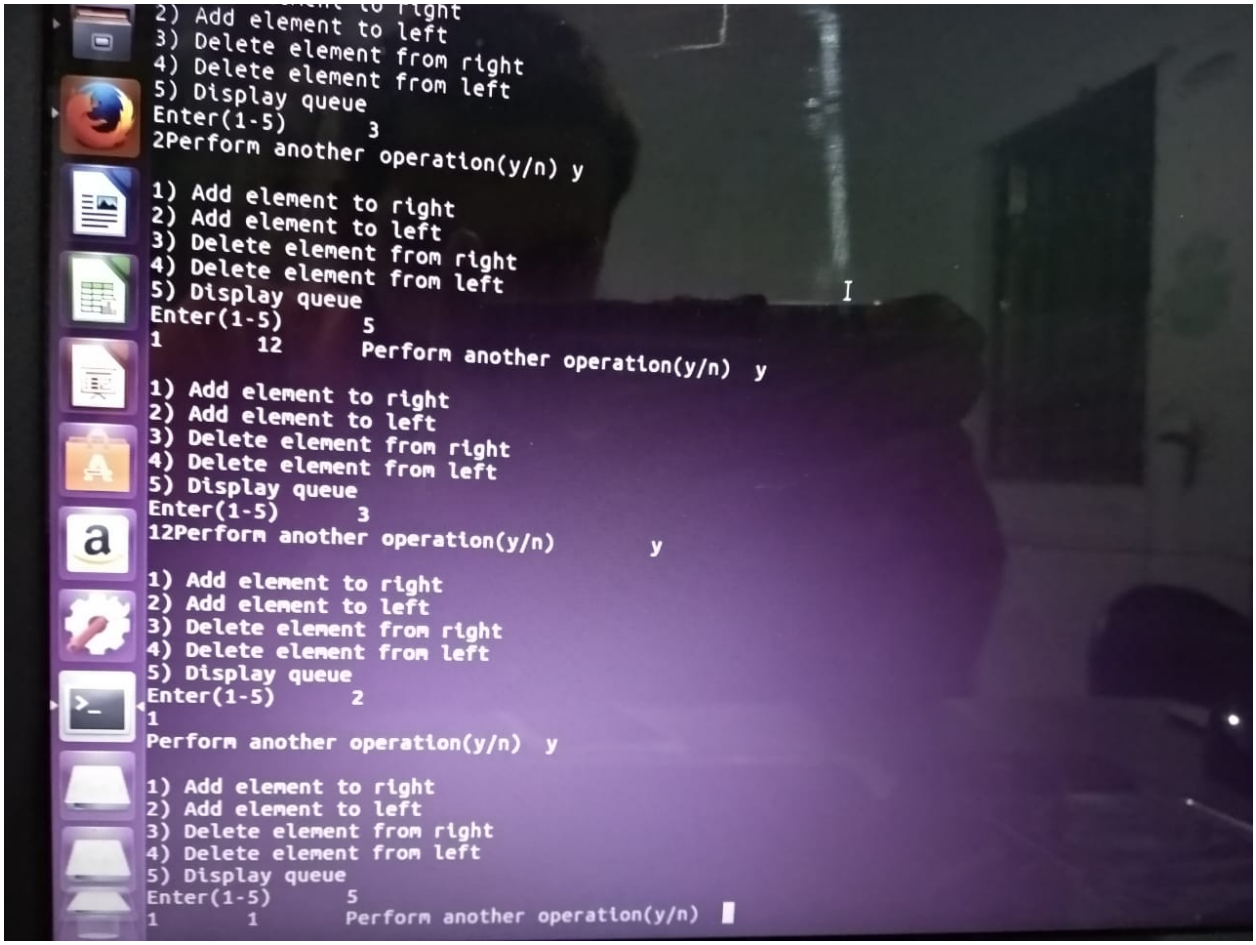
```
#include<stdio.h>
#define mx 3
struct queue{
	int A[mx];
	int front;
	int rear;
	int ne;
};
typedef struct queue Q;
void enqueue(int y, Q * q,char p) {
	int x;
	if(p=='r') {
		if(q->rear%mx!=(q->front+1)%mx)
		{
			q->A[(q->rear++)%mx]=y;
			q->ne++;
```

```
                }
                else if(q->ne==0) {
                        q->A[(q->rear++)%mx]=y;
                        q->ne++;
                }
                else
                printf("\nOverflow\n");
        }
        else if(p=='l'){
                if((q->rear%mx==(q->front+1)%mx)&&q->ne!=0)
        {
                printf("\nOverflow\n");
        }
        else
        {
                if(q->front<0)
                        q->front+=mx;
                q->A[((q->front)--)%mx]=y;
                q->ne++;
        }
        }
}
int dequeue(Q *q,char p) {
        if(p=='l'){
        if(q->rear%mx!=(q->front+1)%mx)
        {
                printf("%d",q->A[(++(q->front))%mx]);
                q->ne--;
        }
        else if(q->ne==0)
        {
                printf("\nqueue empty");
                return;
        }
        else
        {
                printf("%d",q->A[(++(q->front))%mx]);
                q->ne--;
        }
        return (q->A[q->front%mx]);
        }
        else if (p=='r') {
                if(q->rear%mx!=(q->front+1)%mx&&q->ne==0)
```

```
                {
                        printf("\nqueue empty");
                }
                else  {

                        if(q->front<=0)
                        q->front+=mx;
                        printf("%d",q->A[--(q->rear)%mx]);
                        q->ne--;
                        return q->A[(q->rear)%mx];
                }
        }
}
void display(Q *q){
int x=(q->front+1)%mx;
int y=q->ne;
while(y--)
printf("%d\t",q->A[(x++)%mx]);
}
void main(){
Q q;
int ch;
int x;
q.rear=0;
q.front=mx-1;
q.ne=0;
char t='y';
while(t=='y'||t=='Y'){
printf("\n1) Add element to right\n2) Add element to left\n3) Delete element from right\n4) Delete
element from left \n5) Display queue \nEnter(1-5)\t");
scanf("%d",&ch);
switch(ch) {
case 1: scanf("%d",&x);
        enqueue(x,&q,'r');
        break;
case 2: scanf("%d",&x);
        enqueue(x,&q,'l');
        break;
case 3: dequeue(&q,'r');
        break;
case 4: dequeue(&q,'l');
        break;
case 5: display(&q);
```

```
        break;
}
printf("Perform another operation(y/n)\t");
scanf("%c",&t);
scanf("%c",&t);
}
}
```



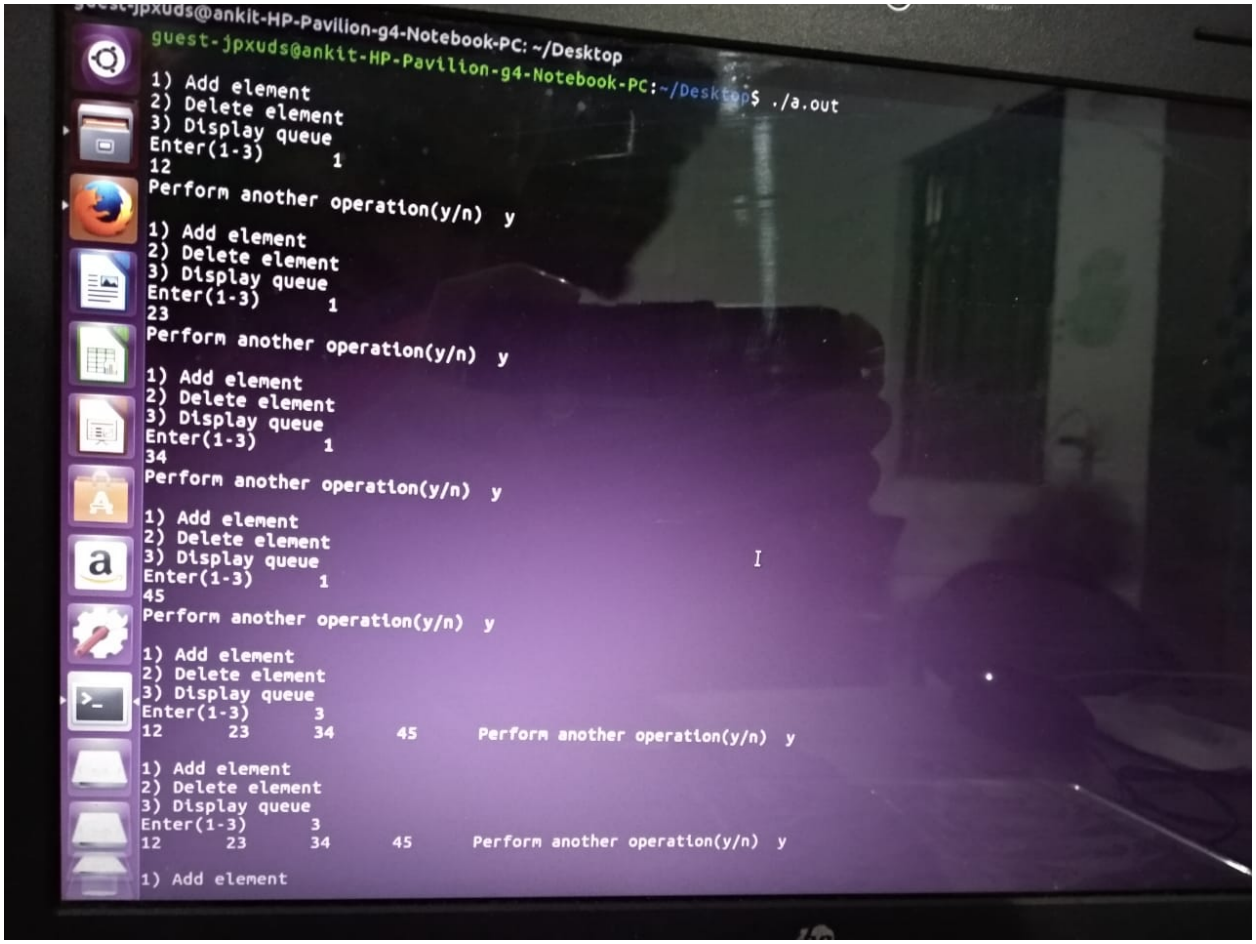
Q4)

```
#include<stdio.h>
#define mx 100
struct queue{
        int A[mx];
        int front;
        int rear;
        int pro[mx];
};
```

```
typedef struct queue Q;
void enqueue(int y, Q * q) {
        int x,i,j;
        if(q->rear!=mx){
        i=q->front+1;
        while(y>q->A[i]&&i!=q->rear)
        i++;
        j=q->rear;
        while(j>i)
        {
   q->A[j]=q->A[j-1];
         j--;
   }
        q->A[i]=y;
        q->rear++;
   //    printf("%d",q->rear);
}
        else if(q->front!=-1) {
        q->front--;
        x=q->front+1;
        while(x!=q->rear-2) {
        q->A[x]=q->A[x+1];
        i=q->front+1;
        while(y>q->A[i]&&i!=q->rear-1)
        i++;
        }
        q->A[i]=y;
        q->rear++;
   //printf("%d",q->rear);
        }
        else
        printf("\nOverflow\n");
}
int dequeue(Q *q) {
if((q->rear)-(q->front)-1==0)
printf("\nqueue empty");
else {
printf("%d",q->A[++(q->front)]);
return (q->A[q->front]);
}
}

void display(Q *q){
```

```
int x=q->front+1;
while(x!=q->rear)
printf("%d\t",q->A[x++]);
}
void main(){
Q q;
int ch;
int x;
q.rear=0;
q.front=-1;
char t='y';
while(t=='y'||t=='Y'){
printf("\n1) Add element\n2) Delete element\n3) Display queue\nEnter(1-3)\t");
scanf("%d",&ch);
switch(ch) {
case 1: scanf("%d",&x);
        enqueue(x,&q);
        break;
case 2: dequeue(&q);
        break;
case 3: display(&q);
}
printf("Perform another operation(y/n)\t");
scanf("%c",&t);
scanf("%c",&t);
}
}
```


guest-jpxuds@ankit-HP-Pavilion-g4-Notebook-PC: ~/Desktop
guest-jpxuds@ankit-HP-Pavilion-g4-Notebook-PC:~/Desktop$ ./a.out
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 1
12
Perform another operation(y/n) y
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 1
23
Perform another operation(y/n) y
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 1
34
Perform another operation(y/n) y
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 1
45
Perform another operation(y/n) y
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 3
12 23 34 45 Perform another operation(y/n) y
1) Add element
2) Delete element
3) Display queue
Enter(1-3) 3
12 23 34 45 Perform another operation(y/n) y
1) Add element