VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

XYZ (1BM19CS000)

Under the Guidance of

Prof. SHEETAL VA
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2020 to Jan-2021

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the LAB RECORD carried out by **XYZ (1BM19CS000)** who is the bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide

Signature of the HOD

Prof. Prof. Sheelal VA
Umadevi V
Assistant Professor
Prof.& Head, Dept. of CSE
BMSCE, Bengaluru
Bengaluru

Dr.
Associate
BMSCE,

External Viva

Name of the Examiner
Signature with date

1._____
_____

2. _____
_____

F Laboratory Plan *(if applicable)*

Lab Program	Unit #	Program Details
1	1	Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow
2	1	WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)
3	2	WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions
4	2	WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions
5	3	WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.
6	3	WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.
7	3	WAP Implement Single Link List with following operations a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists
8	3	WAP to implement Stack & Queues using Linked Representation
9	4	WAP Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value. c) Display the contents of the list
10	5	Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree

```c
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
    //clrscr();
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t--------------------------------");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("\n\t EXIT POINT ");
                break;
            }
            default:
            {
                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
            }

```

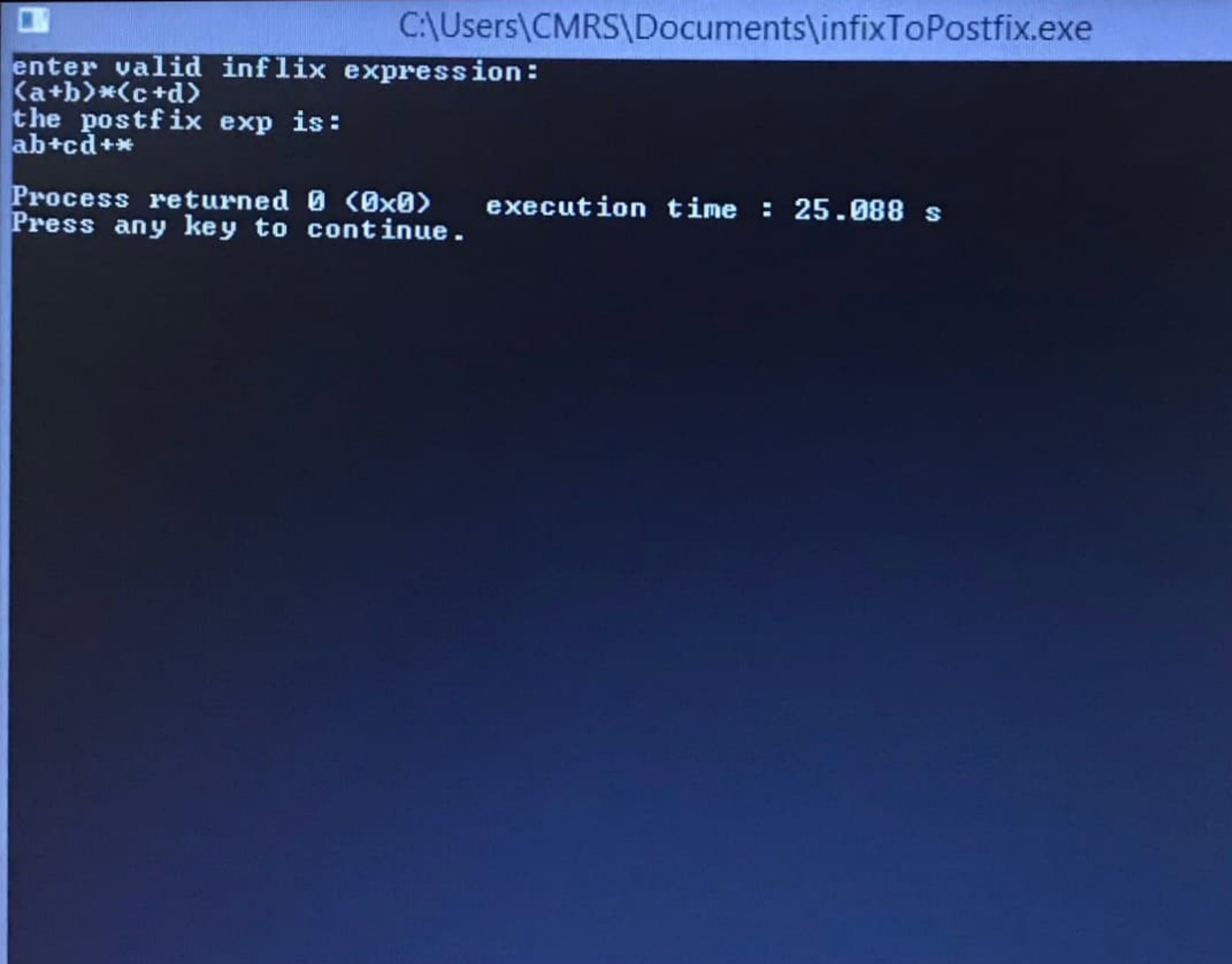
```
    while(choice!=4);
    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");

    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }

}
```

```
Enter the size of STACK[MAX=100]:3

        STACK OPERATIONS USING ARRAY
        ----------------------------

         1.PUSH
         2.POP
         3.DISPLAY
         4.EXIT
Enter the Choice:1
Enter a value to be pushed:20

Enter the Choice:1
Enter a value to be pushed:30

Enter the Choice:1
Enter a value to be pushed:40

Enter the Choice:3

The elements in STACK

40
30
20
 Press Next Choice
Enter the Choice:2

        The popped elements is 40
Enter the Choice:3

The elements in STACK

30
20
 Press Next Choice
Enter the Choice:_
```

```
Enter the Choice:1
Enter a value to be pushed:30

Enter the Choice:1
Enter a value to be pushed:40

Enter the Choice:3

The elements in STACK

40
30
20
Press Next Choice
Enter the Choice:1

      STACK is over flow
Enter the Choice:2

        The popped elements is 40
Enter the Choice:2

        The popped elements is 30
Enter the Choice:2

        The popped elements is 20
Enter the Choice:2

        Stack is under flow
Enter the Choice:_
```

```c
#include<stdio.h>
#include<string.h>
#include<process.h>
int F(char symbol)
{
switch(symbol)
{
case'+':
case'-':return 2;
case'*':
case'/':return 4;
case'^':
case'$':return 5;
case'(':return 0;
case'#':return -1;
default:return 8;
}
}
int G(char symbol)
{
switch(symbol)
{
case'+':
case'-':return 1;
case'*':
case'/':return 3;
case'^':
case'$':return 6;
case'(':return 9;
case')':return 0;
default:return 7;
}
}
void infix_postfix(char infix[],char postfix[])
{
int top,i,j;
char s[30],symbol;
top=-1;
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
{
```

```
case'*':
case'/':return 3;
case'^':
case'$':return 6;
case'(':return 9;
case')':return 0;
default:return 7;
}
}
void infix_postfix(char infix[],char postfix[])
{
int top,i,j;
char s[30],symbol;
top=-1;
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
{
postfix[j]=s[top--];
j++;
}
if(F(s[top])!=G(symbol))
s[++top]=symbol;
else
top--;
}
while(s[top]!='#')
{
postfix[j++]=s[top--];
}
postfix[j]='\0';
}
void main()
{
char infix[20];
char postfix[20];
printf("enter valid inflix expression:\n");
scanf("%s",infix);
infix_postfix(infix,postfix);
printf("the postfix exp is:\n");
printf("%s\n",postfix);
```


C:\Users\CMRS\Documents\infixToPostfix.exe
enter valid inflix expression:
(a+b)*(c+d)
the postfix exp is:
ab+cd+*
Process returned 0 (0x0)   execution time : 25.088 s
Press any key to continue.

```c
#include<stdio.h>
#include<process.h>
#define SIZE 10
void enQueue(int);
void deQueue();
void display();

int queue[SIZE], front = -1, rear = -1;

void main()
{
   int value, choice;

   while(1){
      printf("\n\n***** MENU *****\n");
      printf("1. Insertion\n2. Deletion\n3. Display\n4. Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&choice);
      switch(choice){
case 1: printf("Enter the value to be insert: ");
scanf("%d",&value);
enQueue(value);
break;
case 2: deQueue();
break;
case 3: display();
break;
case 4: exit(0);
default: printf("\nWrong selection!!! Try again!!!");
      }
   }
}
void enQueue(int value){
   if(rear == SIZE-1)
      printf("\nQueue is Full!!! Insertion is not possible!!!");
   else{
      if(front == -1)
front = 0;
      rear++;
      queue[rear] = value;
      printf("\nInsertion success!!!");
   }
}
```

```c
case 1: printf("Enter the value to be insert: ");
scanf("%d",&value);
enQueue(value);
break;
case 2: deQueue();
break;
case 3: display();
break;
case 4: exit(0);
default: printf("\nWrong selection!!! Try again!!!");
      }
   }
}
void enQueue(int value){
   if(rear == SIZE-1)
      printf("\nQueue is Full!!! Insertion is not possible!!!");
   else{
      if(front == -1)
front = 0;
      rear++;
      queue[rear] = value;
      printf("\nInsertion success!!!");
   }
}
void deQueue(){
   if(front == rear)
      printf("\nQueue is Empty!!! Deletion is not possible!!!");
   else{
      printf("\nDeleted : %d", queue[front]);
      front++;
      if(front == rear)
front = rear = -1;
   }
}
void display(){
   if(rear == -1)
      printf("\nQueue is Empty!!!");
   else{
      int i;
      printf("\nQueue elements are:\n");
      for(i=front; i<=rear; i++)
 printf("%d\t",queue[i]);
   }
}
```

```
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 20

Insertion success!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 30

Insertion success!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 40

Insertion success!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 3

Queue elements are:
20      30      40

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 2

Deleted : 20
```

```
Insertion success!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 40

Insertion success!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 50

Queue is Full!!! Insertion is not possible!!!

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 2

Deleted : 20

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 2

Deleted : 30

***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 2

Queue is Empty!!! Deletion is not possible!!!
```

```c
#include <stdio.h>
#define size 3
void insertq(int[], int);
void deleteq(int[]);
void display(int[]);

int front =  - 1;
int rear =  - 1;

int main()
{
    int n, ch;
    int queue[size];
    do
    {
        printf("\n\n Circular Queue:\n1. Insert \n2. Delete\n3. Display\n4. Exit");
printf("\n enter your choice:");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                printf("\nEnter number: ");
                scanf("%d", &n);
                insertq(queue, n);
                break;
            case 2:
                deleteq(queue);
                break;
            case 3:
                display(queue);
                break;
        }
    }while (ch != 0);
}


void insertq(int queue[], int item)
{
    if ((front == 0 && rear == size - 1) || (front == rear + 1))
    {
        printf("queue is full");
        return;
    }
    else if (rear ==  - 1)
    {
```

```
53      else
54      {
55          rear++;
56      }
57      queue[rear] = item;
58  }
59
60  void display(int queue[])
61  {
62      int i;
63      printf("\n");
64      if (front > rear)
65      {
66          for (i = front; i < size; i++)
67          {
68              printf("%d ", queue[i]);
69          }
70          for (i = 0; i <= rear; i++)
71              printf("%d ", queue[i]);
72      }
73      else
74      {
75          for (i = front; i <= rear; i++)
76              printf("%d ", queue[i]);
77      }
78  }
79
80  void deleteq(int queue[])
81  {
82      if (front ==  - 1)
83      {
84          printf("Queue is empty ");
85      }
86      else if (front == rear)
87      {
88          printf("\n %d deleted", queue[front]);
89          front =  - 1;
90          rear =  - 1;
91      }
92      else
93      {
94          printf("\n %d deleted", queue[front]);
95          front++;
96      }
```

```
 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:1

Enter number: 20


 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:1

Enter number: 30


 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:1

Enter number: 40


 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:1
```

```
 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:1

Enter number: 50
queue is full

 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:2

 20 deleted

 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:2

 30 deleted

 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:2

 40 deleted

 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:2
Queue is empty

 Circular Queue:
1. Insert
2. Delete
3. Display
4. Exit
 enter your choice:2
Queue is empty
```

```c
#include<stdio.h>
#include<conio.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("mem 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 IF(NODE second,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
return temp;
```

```
{
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 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 IR(NODE second,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
 return temp;
cur=second;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return second;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
```

```
105   prev=NULL;
106   cur=first;
107   while(cur->link!=NULL)
108   {
109   prev=cur;
110   cur=cur->link;
111   }
112   printf("iten deleted at rear-end is %d",cur->info);
113   free(cur);
114   prev->link=NULL;
115   return first;
116   }
117   NODE insert_pos(int item,int pos,NODE first)
118   {
119   NODE temp;
120   NODE prev,cur;
121   int count;
122   temp=getnode();
123   temp->info=item;
124   temp->link=NULL;
125   if(first==NULL && pos==1)
126   return temp;
127   if(first==NULL)
128   {
129    printf("invalid pos\n");
130    return first;
131   }
132   if(pos==1)
133   {
134   temp->link=first;
135   return temp;
136   }
137   count=1;
138   prev=NULL;
139   cur=first;
140   while(cur!=NULL && count!=pos)
141   {
142    prev=cur;
143    cur=cur->link;
144    count++;
145   }
146   if(count==pos)
147   {
148  prev->link=temp;
```

```
160  if(first==NULL || pos<=0)
161  {
162  printf("invalid position \n");
163  return NULL;
164  }
165  if (pos==1)
166  {
167  cur=first;
168   first=first->link;
169  freenode(cur);
170  return first;
171  }
172  prev=NULL;
173  cur=first;
174  count=1;
175  while(cur!=NULL)
176  {
177  if(count==pos)
178  break;  //if found
179  prev=cur;
180  cur=cur->link;
181  count++;
182  }
183  if(count!=pos)
184  {
185    printf("invalid position\n");
186     return first;
187  }
188  if(count!=pos)
189  {
190    printf("invalid position specified\n");
191    return first;
192  }
193
194  prev->link=cur->link;
195   freenode(cur);
196  return first;
197  }
198  NODE reverse(NODE first)
199    {
200      NODE cur,temp;
201      cur=NULL;
202      while(first!=NULL)
203      {
```

```
215         int temp;
216
217 if(first== NULL) {
218   return 0;
219   }
220 else {
221  while(prev!= NULL) {
222
223   cur = prev->link;
224
225   while(cur!= NULL) {
226
227         if(prev->info > cur->info) {
228   temp = prev->info;
229   prev->info = cur->info;
230   cur->info = temp;
231    }
232    cur = cur->link;
233   }
234   prev= prev->link;
235             }
236        }
237        return first;
238    }
239 NODE des(NODE first)
240 {
241     NODE prev=first;
242     NODE cur=NULL;
243        int temp;
244
245 if(first==NULL) {
246 return 0;
247 }
248 else {
249  while(prev!= NULL) {
250
251  cur = prev->link;
252
253   while(cur!= NULL) {
254
255         if(prev->info < cur->info) {
256   temp = prev->info;
257   prev->info = cur->info;
258   cur->info = temp;
```

```
280         cur->link=second;
281         return first;
282     }
283
284     void display(NODE first)
285     {
286      NODE temp;
287      if(first==NULL)
288      printf("list empty cannot display items\n");
289      for(temp=first;temp!=NULL;temp=temp->link)
290       {
291       printf("%d\n",temp->info);
292       }
293     }
294     void main()
295     {
296     int item,choice,pos,element,option,choice2,item1,num;
297     NODE first=NULL;
298     NODE second=NULL;
299
300     for(;;)
301     {
302     printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:reverse\n 7:sort\n 8.
303     printf("enter the choice\n");
304     scanf("%d",&choice);
305     switch(choice)
306      {
307       case 1:printf("enter the item at front-end\n");
308     scanf("%d",&item);
309     first=insert_front(first,item);
310     break;
311       case 2:first=delete_front(first);
312     break;
313       case 3:printf("enter the item at rear-end\n");
314     scanf("%d",&item);
315     first=insert_rear(first,item);
316     break;
317       case 4:first=delete_rear(first);
318     break;
319      case 5:
320      printf("press 1 to insert or 2 to delete at any desired position \n");
321      scanf("%d",&element);
322      if(element==1){
323              printf("enter the position to insert \n");
```

```
break;
  case 6:
        first=reverse(first);
        break;
  case 7:
        printf("press 1 for ascending sort and 2 for descending sort:\n");
        scanf("%d",&option);
        if(option==1)
        first=asc(first);
        if(option==2)
        first=des(first);
        break;
  case 8:
        printf("create a second list\n");
        printf("enter the number of elements in second list\n");

        scanf("%d",&num);
         for(int i=1;i<=num;i++){
        printf("\n press 1 to insert front and 2 to insert rear \n");
        scanf("%d",&choice2);

            if(choice2==1){
            printf("enter the item at front-end\n");
      scanf("%d",&item1);
          second=IF(second,item1);
           }

          if(choice2==2){
          printf("enter the item at rear-end\n");
     scanf("%d",&item1);
          second=IR(second,item1);
          }
       }

       first=concate(first,second);
       break;

  case 9:display(first);
break;
 default:exit(0);
break;
 }
}
}
```