9.1.1

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    int data;
    struct node *next;
};
struct node *root = NULL;
void insert()
{
    struct node *temp,*p;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter an element: ");
    scanf("%d",&temp->data);
    temp->prev = NULL;
    temp->next = NULL;
    if(root==NULL)
    {
        root = temp;
        root->next = root;
        root->prev = root;
    }
    else
    {
        p = root;
    while(p->next!=root)
    {
        p = p->next;
```

```
30      }
31      p->next = temp;
32      temp->prev = p;
33      temp->next = root;
34      root = temp;
35
36
37      }
38
39    }
40    void delete()
41    {
42      struct node *p,*q;
43
44      if(root==NULL)
45      {
46        printf("Double Linked List is empty so deletion is not
      possible\n");
47      }
48      else
49      {
50      p = root;
51      q = root;
52      while(q->next!=root)
53      {
54        q = q->next;
55      }
56      printf("The deleted element from DCLL : %d\n",p->data);
57      root = root->next;
58      q->next = root;
```

```
59      root->prev = q;
60      p->next = NULL;
61      p->prev = NULL;
62      free(p);
63
64      }
65      }
66      void search()
67      {
68      struct node *p,*q;
69      int kele;
70      int flag = 0;
71      int pos =1;
72      printf("Enter search element: ");
73      scanf("%d",&kele);
74      p = root;
75
76      if(root==NULL)
77      {
78      printf("The given element %d is not found in the given
        DCLL\n",kele);
79      }
80
81
82      else{
83
84      do
85      {
86      if(p->data==kele)
87      {
```

```
88      flag = 1;
89      printf("The given element %d is found at position:
        %d\n",kele,pos);
90      }
91
92      p = p->next;
93
94      pos = pos+1;
95
96    }while(p!=root);
97    }
98    if(flag==0&&root!=NULL)
99    {
100     printf("The given element %d is not found in the given
        DCLL\n",kele);
101   }
102
103   }
104   void traverse()
105   {
106   struct node *p;
107   p = root;
108   if(p==NULL)
109   {
110     printf("Doubly Circular Linked List is empty\n");
111   }
112   else
113   {
114   printf("The elements in DCLL are: ");
```

```
115     
116     while(p->next!=root)
117     {
118     printf("%d ",p->data);
119     p = p->next;
120     }
121     printf("%d ",p->data);
122     printf("\n");
123     }
124     }
125     int main()
126     {
127     int op;
128     while(1)
129     {
130     printf("1.Insert At Begin 2.Delete at Begin 3.Search an element
        Position 4.Traverse the List 5.Exit\n");
131     printf("Enter your option: ");
132     scanf("%d",&op);
133     switch(op)
134     {
135     case 1:
136     insert();
137     break;
138     case 2:
139     delete();
140     break;
141     
142     case 3:
```

```
143     search();
144     break;
145     
146     case 4:
147     traverse();
148     break;
149     
150     case 5:
151     exit(0);
152     }
153     }
154     return 0;
155     
156     }
```

9.1.2

```c
#include<stdio.h>

struct node
{
    int data;
    struct node *next;
};
struct node *top=NULL;

void push(int x)
{
struct node *temp,*p;
temp = (struct node*)malloc(sizeof(struct node));
temp->data = x;
temp->next = NULL;
if(top==NULL)
{
    top = temp;
    printf("Successfully pushed.\n");
}
else
{
    temp->next = top;
    top = temp;
    printf("Successfully pushed.\n");
}
}
void pop()
{
```

```
30      struct node *p;
31      if(top==NULL)
32      {
33          printf("Stack is underflow.\n");
34      }
35      else
36      {
37          p = top;
38          printf("Popped value = %d\n",p->data);
39          top = top->next;
40          p->next = NULL;
41          free(p);
42      }
43  }
44
45  void display()
46  {
47      struct node *p;
48      if(top == NULL)
49      {
50          printf("Stack is empty.\n");
51
52      }
53      else
54      {
55          p = top;
56          printf("Elements of the stack are : ");
57          while(p!=NULL)
58          {
59              printf("%d ",p->data);
```

```
60          p = p->next;
61      }
62      printf("\n");
63  }
64  }
65  void peek()
66  {
67      struct node *p;
68      if(top==NULL)\
69      {
70          printf("Stack is underflow.\n");
71      }
72      else
73      {
74          p = top;
75          printf("Peek value = %d\n",p->data);
76
77      }
78  }
79
80  void isEmpty()
81  {
82      struct node *p;
83      if(top==NULL)
84      {
85          printf("Stack is empty.\n");
86      }
87      else
88      {
89          printf("Stack is not empty.\n");
```

```
90      }
91  }
92
```

9.1.3

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

typedef struct node {
int data;
struct node* next;
} Node;

Node* front = NULL;
Node* rear = NULL;
int siz = 0;

void enqueue(int data) {
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->data = data;
    temp->next = NULL;
    if (front == NULL && rear == NULL) {
        front = rear = temp;
    } else {
        rear->next = temp;
        rear = temp;
    }
    siz++;
    printf("Successfully inserted.\n");
}
void dequeue() {
    if (front == NULL) {
        printf("Queue is underflow.\n");
    } else {
```

```
        Node* temp = front;
        front = front->next;
        if (front == NULL) {
            rear = NULL;
        }
        printf("Deleted value = %d\n", temp->data);
        free(temp);
        siz--;
    }
}
void display() {
    if (front == NULL) {
        printf("Queue is empty.\n");
    } else {
        Node* temp = front;
        printf("Elements in the queue : ");
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}

void isEmpty() {
    if (front == NULL) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue is not empty.\n");
```

```
    }
}

void size() {
    printf("Queue size : %d\n", siz);
}

```

9.2.1

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

struct student {
char name[50];
int grade;
} Student;

void bubble_sort(struct student arr[], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i].grade > arr[j].grade) {
                struct student temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
int main() {
    int n;
    printf("Enter the number of students: ");
    scanf("%d", &n);
    struct student students[n];
    for (int i = 0; i < n; i++) {
        printf("Enter name of student %d: ", i + 1);
        scanf("%s", students[i].name);
        printf("Enter grade of student %d: ", i + 1);
        scanf("%d", &students[i].grade);
```

```c
    }
    bubble_sort(students, n);
    printf("Sorted students by grade: \n");
    for (int i = 0; i < n; i++) {
        printf("%s: %d\n", students[i].name, students[i].grade);
    }

}
```