Assignment 6

Answer 1(a):

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

struct Node{
   int data;
   struct Node* next;
};
struct Node* head=NULL;

struct Node* create(int x){
   struct Node* p=malloc(sizeof(struct Node));
   p->data=x;
   p->next=NULL;
   return p;
}

void insert_begin(int x){
   struct Node* p=create(x);
   if(head==NULL){
      head=p;
      p->next=p;
      return;
   }
   struct Node* t=head;
   while(t->next!=head) t=t->next;
   p->next=head;
   t->next=p;
   head=p;
}

void insert_end(int x){
   struct Node* p=create(x);
   if(head==NULL){
      head=p;
      p->next=p;
      return;
   }
   struct Node* t=head;
   while(t->next!=head) t=t->next;
```

```
    t->next=p;
    p->next=head;
}

int insert_before(int key,int x){
    if(head==NULL) return 0;
    if(head->data==key){
        insert_begin(x);
        return 1;
    }
    struct Node* prev=head;
    struct Node* cur=head->next;
    while(cur!=head){
        if(cur->data==key){
            struct Node* p=create(x);
            prev->next=p;
            p->next=cur;
            return 1;
        }
        prev=cur;
        cur=cur->next;
    }
    return 0;
}

int insert_after(int key,int x){
    if(head==NULL) return 0;
    struct Node* cur=head;
    do{
        if(cur->data==key){
            struct Node* p=create(x);
            p->next=cur->next;
            cur->next=p;
            return 1;
        }
        cur=cur->next;
    }while(cur!=head);
    return 0;
}

int delete_node(int key){
    if(head==NULL) return 0;
    if(head->data==key){
        if(head->next==head){
```

```
            free(head);
            head=NULL;
            return 1;
        }
        struct Node* t=head;
        while(t->next!=head) t=t->next;
        t->next=head->next;
        struct Node* d=head;
        head=head->next;
        free(d);
        return 1;
    }
    struct Node* prev=head;
    struct Node* cur=head->next;
    while(cur!=head){
        if(cur->data==key){
            prev->next=cur->next;
            free(cur);
            return 1;
        }
        prev=cur;
        cur=cur->next;
    }
    return 0;
}

int search(int key){
    if(head==NULL) return 0;
    struct Node* cur=head;
    do{
        if(cur->data==key) return 1;
        cur=cur->next;
    }while(cur!=head);
    return 0;
}

void display(){
    if(head==NULL){
        printf("Empty\n");
        return;
    }
    struct Node* t=head;
    do{
        printf("%d ",t->data);
```

```
        t=t->next;
    }while(t!=head);
    printf("\n");
}

int main(){
    int ch,x,k;
    while(1){
        scanf("%d",&ch);
        if(ch==0) break;
        if(ch==1){ scanf("%d",&x); insert_begin(x); }
        else if(ch==2){ scanf("%d",&x); insert_end(x); }
        else if(ch==3){ scanf("%d %d",&k,&x); insert_before(k,x); }
        else if(ch==4){ scanf("%d %d",&k,&x); insert_after(k,x); }
        else if(ch==5){ scanf("%d",&k); delete_node(k); }
        else if(ch==6){ scanf("%d",&k); printf(search(k)?"Found\n":"Not Found\n"); }
        else if(ch==7) display();
    }
    return 0;
}
```

Answer 1(b):

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

struct Node{
    int data;
    struct Node *prev,*next;
};
struct Node* head=NULL;

struct Node* create(int x){
    struct Node* p=malloc(sizeof(struct Node));
    p->data=x;
    p->prev=NULL;
    p->next=NULL;
    return p;
}

void insert_begin(int x){
    struct Node* p=create(x);
    if(head==NULL){
        head=p;
```

```
        return;
    }
    p->next=head;
    head->prev=p;
    head=p;
}

void insert_end(int x){
    struct Node* p=create(x);
    if(head==NULL){
        head=p;
        return;
    }
    struct Node* t=head;
    while(t->next) t=t->next;
    t->next=p;
    p->prev=t;
}

int insert_before(int key,int x){
    if(head==NULL) return 0;
    if(head->data==key){
        insert_begin(x);
        return 1;
    }
    struct Node* t=head;
    while(t){
        if(t->data==key){
            struct Node* p=create(x);
            p->next=t;
            p->prev=t->prev;
            t->prev->next=p;
            t->prev=p;
            return 1;
        }
        t=t->next;
    }
    return 0;
}

int insert_after(int key,int x){
    struct Node* t=head;
    while(t){
        if(t->data==key){
```

```
            struct Node* p=create(x);
            p->next=t->next;
            p->prev=t;
            if(t->next) t->next->prev=p;
            t->next=p;
            return 1;
        }
        t=t->next;
    }
    return 0;
}

int delete_node(int key){
    if(head==NULL) return 0;
    if(head->data==key){
        struct Node* d=head;
        head=head->next;
        if(head) head->prev=NULL;
        free(d);
        return 1;
    }
    struct Node* t=head;
    while(t){
        if(t->data==key){
            if(t->prev) t->prev->next=t->next;
            if(t->next) t->next->prev=t->prev;
            free(t);
            return 1;
        }
        t=t->next;
    }
    return 0;
}

int search(int key){
    struct Node* t=head;
    while(t){
        if(t->data==key) return 1;
        t=t->next;
    }
    return 0;
}

void display(){
```

```
    struct Node* t=head;
    while(t){
        printf("%d ",t->data);
        t=t->next;
    }
    printf("\n");
}

int main(){
    return 0;
}
```

Answer 2:

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

struct Node{int data; struct Node* next;};
struct Node* head=NULL;

int main(){
    struct Node* a=malloc(sizeof(struct Node));
    struct Node* b=malloc(sizeof(struct Node));
    struct Node* c=malloc(sizeof(struct Node));
    struct Node* d=malloc(sizeof(struct Node));
    struct Node* e=malloc(sizeof(struct Node));
    a->data=20; b->data=100; c->data=40; d->data=80; e->data=60;
    a->next=b; b->next=c; c->next=d; d->next=e; e->next=a;
    head=a;

    struct Node* t=head;
    do{
        printf("%d ",t->data);
        t=t->next;
    }while(t!=head);
    printf("%d\n",head->data);
    return 0;
}
```

Answer 3 (doubly):

```
int size_dll(struct Node* head){
    int c=0;
    while(head){ c++; head=head->next; }
```

```
    return c;
}
```

Answer 3(circular):

```
int size_cll(struct Node* head){
    if(head==NULL) return 0;
    int c=0;
    struct Node* t=head;
    do{
        c++;
        t=t->next;
    }while(t!=head);
    return c;
}
```

Answer 4:

```
int isPalindrome(struct Node* head){
    if(head==NULL) return 1;
    struct Node* tail=head;
    while(tail->next) tail=tail->next;
    while(head!=tail && tail->next!=head){
        if(head->data!=tail->data) return 0;
        head=head->next;
        tail=tail->prev;
    }
    return 1;
}
```

Answer 5:

```
int isCircular(struct Node* head){
    if(head==NULL) return 1;
    struct Node* t=head->next;
    while(t && t!=head) t=t->next;
    return t==head;
}
```