Assignment 5

Answer 1:

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

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

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

void insert_begin(struct Node** head,int x){struct Node* p=create(x);p->next=*head;*head=p;}
void insert_end(struct Node** head,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;}
int insert_before(struct Node** head,int key,int x){
   if(*head==NULL) return 0;
   if((*head)->data==key){insert_begin(head,x);return 1;}
   struct Node* prev=*head; struct Node* cur=(*head)->next;
     while(cur){ 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(struct Node** head,int key,int x){
   struct Node* cur=*head;
         while(cur){ if(cur->data==key){ struct Node* p=create(x); p->next=cur->next; cur->next=p; return 1;} cur=cur->next;}
   return 0;
}
int delete_begin(struct Node** head){
   if(*head==NULL) return 0;
   struct Node* t=*head; *head=(*head)->next; free(t); return 1;
}
int delete_end(struct Node** head){
   if(*head==NULL) return 0;
   if((*head)->next==NULL){ free(*head); *head=NULL; return 1;}
      struct Node* t=*head; while(t->next->next) t=t->next; free(t->next); t->next=NULL; return 1;
}
int delete_key(struct Node** head,int key){
   if(*head==NULL) return 0;
   if((*head)->data==key){ struct Node* t=*head; *head=(*head)->next; free(t); return 1;}
   struct Node* prev=*head; struct Node* cur=(*head)->next;
     while(cur){ if(cur->data==key){ prev->next=cur->next; free(cur); return 1;} prev=cur;
cur=cur->next;}
   return 0;
}
int search_pos(struct Node* head,int key){
   int pos=1; while(head){ if(head->data==key) return pos; head=head->next; pos++; } return
-1;
}
void display(struct Node* head){ if(head==NULL){ printf("List is empty\n"); return;}
while(head){ printf("%d",head->data); if(head->next) printf("->"); head=head->next;}
printf("\n");}

int main(){
   struct Node* head=NULL;
   int choice,x,key,ans;
   while(1){
         printf("1.Insert at beginning\n2.Insert at end\n3.Insert before a value\n4.Insert after a
value\n5.Delete from beginning\n6.Delete from end\n7.Delete specific node\n8.Search
node\n9.Display\n0.Exit\n");
       if(scanf("%d",&choice)!=1) return 0;
       switch(choice){
          case 1: scanf("%d",&x); insert_begin(&head,x); break;
          case 2: scanf("%d",&x); insert_end(&head,x); break;
                   case 3: scanf("%d %d",&key,&x); ans=insert_before(&head,key,x); if(!ans)
printf("Value not found\n"); break;
                     case 4: scanf("%d %d",&key,&x); ans=insert_after(&head,key,x); if(!ans)
printf("Value not found\n"); break;
          case 5: if(!delete_begin(&head)) printf("List empty\n"); break;
          case 6: if(!delete_end(&head)) printf("List empty\n"); break;
             case 7: scanf("%d",&key); if(!delete_key(&head,key)) printf("Value not found\n");
break;
                 case 8: scanf("%d",&key); ans=search_pos(head,key); if(ans==-1) printf("Not
found\n"); else printf("Position: %d\n",ans); break;
          case 9: display(head); break;
          case 0: return 0;
          default: printf("Invalid\n");
       }
   }
   return 0;
}
```

Answer 2:

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

struct Node{int data; struct Node* next;};
struct Node* create(int x){struct Node* p=(struct Node*)malloc(sizeof(struct Node));p->data=x;p->next=NULL;return p;}
void insert_end(struct Node** head,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;}
int count_key(struct Node* head,int key){int c=0; while(head){ if(head->data==key) c++; head=head->next;} return c;}
void delete_all(struct Node** head,int key){
      while(*head && (*head)->data==key){ struct Node* t=*head; *head=(*head)->next; free(t);}
   if(*head==NULL) return;
   struct Node* prev=*head; struct Node* cur=(*head)->next;
   while(cur){
      if(cur->data==key){ prev->next=cur->next; free(cur); cur=prev->next; }
      else { prev=cur; cur=cur->next; }
   }
}
void display(struct Node* head){ if(head==NULL){ printf("Empty\n"); return;} while(head){ printf("%d",head->data); if(head->next) printf("->"); head=head->next;} printf("\n");}

int main(){
   int arr[]={1,2,1,2,1,3,1}; int n=7;
   struct Node* head=NULL;
   for(int i=0;i<n;i++) insert_end(&head,arr[i]);
   int key=1;
   int c=count_key(head,key);
   printf("Count: %d\n",c);
   delete_all(&head,key);
   printf("Updated Linked List: ");
   display(head);
   return 0;
}
```

Answer 3:

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

struct Node{int data; struct Node* next;};
struct Node* create(int x){struct Node* p=(struct Node*)malloc(sizeof(struct
Node));p->data=x;p->next=NULL;return p;}
void insert_end(struct Node** head,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;}
int find_middle(struct Node* head){
   struct Node* slow=head; struct Node* fast=head;
   while(fast && fast->next){ slow=slow->next; fast=fast->next->next; }
   if(slow) return slow->data; return -1;
}
int main(){
   struct Node* head=NULL;
      insert_end(&head,1); insert_end(&head,2); insert_end(&head,3); insert_end(&head,4);
insert_end(&head,5);
   printf("%d\n",find_middle(head));
   return 0;
}
```

Answer 4:

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

struct Node{int data; struct Node* next;};
struct Node* create(int x){struct Node* p=(struct Node*)malloc(sizeof(struct
Node));p->data=x;p->next=NULL;return p;}
void insert_end(struct Node** head,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;}
struct Node* reverse(struct Node* head){
   struct Node* prev=NULL; struct Node* cur=head; struct Node* nxt;
   while(cur){ nxt=cur->next; cur->next=prev; prev=cur; cur=nxt; }
   return prev;
}
void display(struct Node* head){ if(head==NULL){ printf("NULL\n"); return;} while(head){
printf("%d",head->data); if(head->next) printf("->"); head=head->next;}
printf("->NULL\n");}
int main(){
   struct Node* head=NULL;
   insert_end(&head,1); insert_end(&head,2); insert_end(&head,3); insert_end(&head,4);
   head=reverse(head);
   display(head);
   return 0;
}
```