DSA

FISAC - 1

NAYAKANTI SAI MIHIRNATH
210905368
A - 57

Contents

List of Figures

LinkedList.c

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

typedef struct node *Nodeptr;

struct node
{
  int data;
  Nodeptr next;
};

Nodeptr getnode()
{
  Nodeptr temp;
  temp = (Nodeptr)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("No Free Space");
    return NULL;
  }
  return temp;
}

int isEmpty(Nodeptr list)
{
  if(list==NULL)
  {
    return 1;
  }
  return 0;
}

Nodeptr InsertFront(Nodeptr first,int item)
{
  Nodeptr temp;
  temp = getnode();
  temp->data=item;
  temp->next=first;
  return temp;
}

Nodeptr InsertLast(Nodeptr first,int item)
{
  Nodeptr temp,rear;
  temp=getnode();
  rear=getnode();
  temp->data = item;
  temp->next = NULL;
  rear=first;
  if(isEmpty(first))
  {
    first = temp;
  }
  else
  {
    while(rear->next)
    {
      rear=rear->next;
    }
    rear->next=temp;
  }
  return first;
}

int DeleteFront(Nodeptr *first)
{
  Nodeptr temp;
  int x;
  temp = *first;
  if(isEmpty(*first))
  {
    printf("List is Empty");
    return -1;
  }
```

```
    else
    {
      *first = (*first)->next;
      x=temp->data;
      free(temp);
      return x;
    }
  }

  int DeleteLast(Nodeptr *first)
  {
    Nodeptr prev,temp;
    int x;
    if(isEmpty(*first))
    {
      printf("Empty List");
      return -1;
    }
    else
    {
      prev=NULL;
      temp = *first;
      while(temp->next)
      {
        prev=temp;
        temp=temp->next;
      }
      x = temp->data;
      if(isEmpty(prev))
      {
        *first = NULL;
      }
      else
      {
        prev->next = NULL;
      }
      free(temp);
      return x;
    }
  }

  Nodeptr sortlist(Nodeptr first)
  {
    Nodeptr temp=first,rear;
    int x;
    if(isEmpty(temp))
    {
      return NULL;
    }
    else
    {
      while(temp)
      {
        rear=temp->next;
        while(rear)
        {
          if(temp->data > rear->data)
          {
            x=temp->data;
            temp->data=rear->data;
            rear->data=x;
          }
          rear=rear->next;
        }
        temp=temp->next;
      }
    }
    return first;
  }

  void DeleteKey(Nodeptr *first, int key)
  {
    Nodeptr temp = *first,prev;
    while (temp != NULL && temp->data == key)
    {
      *first = temp->next;
```

```
      free(temp);
      temp = *first;
    }
    while (temp != NULL)
    {
      while (temp != NULL && temp->data != key)
      {
        prev = temp;
        temp = temp->next;
      }
      if (temp == NULL) return;
      prev->next = temp->next;
      free(temp);
      temp = prev->next;
    }
  }


  void Display(Nodeptr first)
  {
    Nodeptr temp;
    if(isEmpty(first))
    {
      printf("List is Empty");
    }
    else
    {
      temp=first;
      while(temp)
      {
        printf("%d\t",temp->data);
        temp=temp->next;
      }
    }
  }
```

Q1.c

```c
#include <stdio.h>
#include "LinkedList.c"

void main()
{
  Nodeptr first=NULL,second=NULL;
  first=InsertLast(first,2);
  first=InsertLast(first,4);
  first=InsertLast(first,6);
  first=InsertLast(first,8);
  second=InsertLast(second,3);
  second=InsertLast(second,5);
  second=InsertLast(second,7);
  second=InsertLast(second,9);
  printf("\nList 1 : \n");
  Display(first);
  printf("\nList 2 : \n");
  Display(second);
  printf("\nAfter Merge : \n");
  Nodeptr temp = second;
  while(temp)
  {
    first=InsertLast(first,temp->data);
    temp=temp->next;
  }
  first=sortlist(first);
  Display(first);
}
```

Q2.c

```
#include <stdio.h>
#include "LinkedList.c"

void main()
{
  Nodeptr first=NULL;
  Nodeptr temp,left,right;
  int k,x;
  first=InsertLast(first,3);
  first=InsertLast(first,4);
  first=InsertLast(first,5);
  first=InsertLast(first,6);
  first=InsertLast(first,7);
  printf("Initial List is \n");
  Display(first);
  printf("\n\nEnter k : \t");
  scanf("%d",&k);
  int t=2*k;
  temp=first;
  while(k>0)
  {
    x=DeleteFront(&temp);
    left=InsertLast(temp,x);
    k--;
  }
  printf("\nLeft Rotate : \n");
  Display(left);
  right=left;
  while(t>0)
  {
    x=DeleteLast(&right);
    right=InsertFront(right,x);
    t--;
  }
  printf("\nRight Rotate : \n");
  Display(right);
}
```

Q3.c

```c
#include <stdio.h>
#include "LinkedList.c"

void main()
{
  int n,first=0,last=1,count=2,temp;
  Nodeptr list=NULL;
  list=InsertLast(list,first);
  list=InsertLast(list,last);
  printf("Enter Number of Fibonacci Numbers : \t");
  scanf("%d",&n);
  while(count<n)
  { temp=first;
    first=last;
    last=last+temp;
    list=InsertLast(list,last);
    count++;
  }
  printf("Fibonacci Series is \n");
  Display(list);
}
```

Q4.c

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

typedef struct node *Nodeptr;

struct node
{
  char name[100];
  int data1,data2;
  Nodeptr next;
};

Nodeptr getnode()
{
  Nodeptr temp;
  temp = (Nodeptr)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("No Free Space");
    return NULL;
  }
  return temp;
}

int isEmpty(Nodeptr list)
{
  if(list==NULL)
  {
    return 1;
  }
  return 0;
}

Nodeptr InsertLast(Nodeptr first,int item1,int item2,char names[])
{
  Nodeptr temp,rear;
  temp=getnode();
  rear=getnode();
  strcpy(temp->name,names);
  temp->data1=item1;
  temp->data2=item2;
  temp->next = NULL;
  rear=first;
  if(isEmpty(first))
  {
    first = temp;
  }
  else
  {
    while(rear->next)
    {
      rear=rear->next;
    }
    rear->next=temp;
  }
  return first;
}

Nodeptr sortlist(Nodeptr first)
{
  Nodeptr temp=first,rear;
  int x,y;
  char ss[100];
  if(isEmpty(temp))
  {
    return NULL;
  }
  else
  {
    while(temp)
    {
      rear=temp->next;
      while(rear)
```

```
        {
          if((temp->data1 + temp->data2) > (rear->data1 + rear->data2))
          {
            x=temp->data1;
            temp->data1=rear->data1;
            rear->data1=x;

            y=temp->data2;
            temp->data2=rear->data2;
            rear->data2=y;

            strcpy(ss,temp->name);
            strcpy(temp->name,rear->name);
            strcpy(rear->name,ss);
          }
          rear=rear->next;
        }
        temp=temp->next;
      }
    }
    return first;
}

void Display(Nodeptr first)
{
  Nodeptr temp;
  if(isEmpty(first))
  {
    printf("List is Empty");
  }
  else
  {
    temp=first;
    while(temp)
    {
      printf("%s\t%d\t%d\t\n",temp->name,temp->data1,temp->data2);
      temp=temp->next;
    }
  }
}

void main()
{
  Nodeptr first = NULL,temp;
  first=InsertLast(first,12,45,"Arun");
  first=InsertLast(first,6,89,"Ram");
  first=InsertLast(first,1,60,"Raju");
  Display(first);
  printf("\n");
  first=sortlist(first);
  temp=first;
  temp=temp->next;
  printf("\nDetails of Student with Second Highest Marks are \n");
  printf("%s\t%d\t%d",temp->name,temp->data1,temp->data2);
}
```

Q5.c

```c
#include <stdio.h>
#include "LinkedList.c"

void main()
{
  int choice;
  Nodeptr first=NULL;
  printf("Enter Choice :\t");
  scanf("%d",&choice);
  while(choice!=4)
  {
    if(choice==1)
    {
      int x,n;
      printf("Enter Number of Nodes : \t");
      scanf("%d",&n);
      for(int i=0;i<n;i++)
      {
        printf("Enter Data of Node %d : \t",i+1);
        scanf("%d",&x);
        first=InsertLast(first,x);
      }
      printf("\nData in List Before Deletion : \n");
      Display(first);
    }

    if(choice==2)
    {
      int key,keysq,count=0;
      printf("Enter Key : \t");
      scanf("%d",&key);
      keysq=key*key;
      DeleteKey(&first,keysq);
    }
    if(choice==3)
    {
      printf("\nData in List After Deletion : \n");
      Display(first);
    }
    printf("\nEnter Choice :\t");
    scanf("%d",&choice);
  }
}
```

Q6.c

```c
#include <stdio.h>
#include "LinkedList.c"
void main()
{
  Nodeptr first=NULL,list1=NULL,list2=NULL;
  first=InsertLast(first,1);
  first=InsertLast(first,2);
  first=InsertLast(first,3);
  first=InsertLast(first,4);
  first=InsertLast(first,5);
  first=InsertLast(first,6);
  printf("Initial List : \n");
  Display(first);
  int x,y;
  while(first)
  {
    x=first->data;
    y=x*x*x;
    if(x%2==0)
    {
      list1=InsertLast(list1,y);
    }
    if(x%2==1)
    {
      list2=InsertLast(list2,y);
    }
    first=first->next;
  }
  printf("\nList 1 : \n");
  Display(list1);
  printf("\nList 2 : \n");
  Display(list2);
}
```

Q7.c

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

typedef struct node *Nodeptr;

struct node
{
  char data[100];
  Nodeptr next;
};

Nodeptr getnode()
{
  Nodeptr temp;
  temp = (Nodeptr)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("No Free Space");
    return NULL;
  }
  return temp;
}

int isEmpty(Nodeptr list)
{
  if(list==NULL)
  {
    return 1;
  }
  return 0;
}

Nodeptr InsertLast(Nodeptr first,char item[])
{
  Nodeptr temp,rear;
  temp=getnode();
  rear=getnode();
  strcpy(temp->data,item);
  temp->next = NULL;
  rear=first;
  if(isEmpty(first))
  {
    first = temp;
  }
  else
  {
    while(rear->next)
    {
      rear=rear->next;
    }
    rear->next=temp;
  }
  return first;
}

void Display(Nodeptr first)
{
  Nodeptr temp;
  if(isEmpty(first))
  {
    printf("List is Empty");
  }
  else
  {
    temp=first;
    while(temp)
    {
      printf("%s\t",temp->data);
      temp=temp->next;
    }
  }
}
```

```
void main()
{
  Nodeptr first = NULL,PSLIST=NULL,NPSLIST=NULL;
  first=InsertLast(first,"SIRI");
  first=InsertLast(first,"MAM");
  first=InsertLast(first,"lila");
  first=InsertLast(first,"CAC");
  printf("String List is \n");
  Display(first);
  int n,flag;
  char x[100];
  while(first)
  {
    flag=1;
    strcpy(x,first->data);
    n=strlen(x);
    for(int i=0;i<n/2;i++)
    {
      if(x[i]!=x[n-i-1])
      {
        flag=0;
        break;
      }
    }
    if(flag)
    {
      PSLIST=InsertLast(PSLIST,x);
    }
    else
    {
      NPSLIST=InsertLast(NPSLIST,x);
    }
    first=first->next;
  }
  printf("\nPSLIST : \n");
  Display(PSLIST);
  printf("\nNPSLIST : \n");
  Display(NPSLIST);
}
```

Q8.c

```c
#include <stdio.h>
#include "LinkedList.c"

void main()
{
  Nodeptr first=NULL,list1=NULL,list2=NULL;
  first=InsertLast(first,1);
  first=InsertLast(first,2);
  first=InsertLast(first,6);
  first=InsertLast(first,4);
  first=InsertLast(first,8);
  printf("Initial List : \n");
  Display(first);
  int count=0,x;
  while(first)
  {
    count++;
    x=first->data;
    if(count%2==1)
    {
      list1=InsertLast(list1,x);
    }
    else
    {
      list2=InsertLast(list2,x);
    }
    first=first->next;
  }
  printf("\nList 1 : \n");
  Display(list1);
  printf("\nList 2 : \n");
  Display(list2);
}
```

Q9.c

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

typedef struct node *Nodeptr;

struct node
{
  char data;
  Nodeptr next;
};

Nodeptr getnode()
{
  Nodeptr temp;
  temp = (Nodeptr)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("No Free Space");
    return NULL;
  }
  return temp;
}

int isEmpty(Nodeptr list)
{
  if(list==NULL)
  {
    return 1;
  }
  return 0;
}

Nodeptr InsertLast(Nodeptr first,char item)
{
  Nodeptr temp,rear;
  temp=getnode();
  rear=getnode();
  temp->data = item;
  temp->next = NULL;
  rear=first;
  if(isEmpty(first))
  {
    first = temp;
  }
  else
  {
    while(rear->next)
    {
      rear=rear->next;
    }
    rear->next=temp;
  }
  return first;
}

void DeleteKey(Nodeptr *first, char key)
{
  Nodeptr temp = *first,prev;
  while (temp != NULL && temp->data == key)
  {
    *first = temp->next;
    free(temp);
    temp = *first;
  }
  while (temp != NULL)
  {
    while (temp != NULL && temp->data != key)
    {
      prev = temp;
      temp = temp->next;
    }
    if (temp == NULL) return;
    prev->next = temp->next;
```

```
      free(temp);
      temp = prev->next;
    }
  }


  void Display(Nodeptr first)
  {
    Nodeptr temp;
    if(isEmpty(first))
    {
      printf("List is Empty");
    }
    else
    {
      temp=first;
      while(temp)
      {
        printf("%c\t",temp->data);
        temp=temp->next;
      }
    }
  }

  void main()
  {
    Nodeptr first=NULL;
    char str[100];
    printf("Enter Name : \t");
    scanf("%[^\n]",str);
    // gets(str);
    int i=0;
    while(str[i]!='\0')
    {
      first=InsertLast(first,str[i]);
      i++;
    }
    printf("\n");
    Display(first);
    printf("\n");
    printf("Name After removing Vowels : \n");
    DeleteKey(&first,'a');
    DeleteKey(&first,'e');
    DeleteKey(&first,'i');
    DeleteKey(&first,'o');
    DeleteKey(&first,'u');
    DeleteKey(&first,'A');
    DeleteKey(&first,'E');
    DeleteKey(&first,'I');
    DeleteKey(&first,'O');
    DeleteKey(&first,'U');
    printf("\n");
    Display(first);
  }
```

Q10.c

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

typedef struct node *Nodeptr;

struct node
{
  char data[100];
  Nodeptr next;
};

Nodeptr getnode()
{
  Nodeptr temp;
  temp = (Nodeptr)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("No Free Space");
    return NULL;
  }
  return temp;
}

int isEmpty(Nodeptr list)
{
  if(list==NULL)
  {
    return 1;
  }
  return 0;
}

Nodeptr InsertLast(Nodeptr first,char item[])
{
  Nodeptr temp,rear;
  temp=getnode();
  rear=getnode();
  strcpy(temp->data,item);
  temp->next = NULL;
  rear=first;
  if(isEmpty(first))
  {
    first = temp;
  }
  else
  {
    while(rear->next)
    {
      rear=rear->next;
    }
    rear->next=temp;
  }
  return first;
}

void Display(Nodeptr first)
{
  Nodeptr temp;
  if(isEmpty(first))
  {
    printf("List is Empty");
  }
  else
  {
    temp=first;
    while(temp)
    {
      printf("%s\t",temp->data);
      temp=temp->next;
    }
  }
}
```

```
void DeleteKey(Nodeptr *first, char key[])
{
  Nodeptr temp = *first,prev;
  temp =  temp->next;
  while (temp != NULL && strcmp(temp->data,key)==0)
  {
    *first = temp->next;
    free(temp);
    temp = *first;
  }
  while (temp != NULL)
  {
    while (temp != NULL && strcmp(temp->data,key)!=0)
    {
      prev = temp;
      temp = temp->next;
    }
    if (temp == NULL) return;
    prev->next = temp->next;
    free(temp);
    temp = prev->next;
  }
}

int Search(Nodeptr first,char str[])
{
  Nodeptr temp=first;
  int count=0;
  while(temp)
  {
    if(strcmp(str,temp->data)==0)
    {
      count++;
    }
    temp=temp->next;
  }
  return count;
}

void main()
{
  Nodeptr first = NULL;
  first=InsertLast(first,"raj");
  first=InsertLast(first,"Aryan");
  first=InsertLast(first,"Anirudh");
  first=InsertLast(first,"Aryan");
  first=InsertLast(first,"prasad");
  first=InsertLast(first,"Aryan");
  first=InsertLast(first,"Anirudh");
  printf("Initial List \n");
  Display(first);
  Nodeptr temp = first;
  printf("\n");
  while(temp)
  {
    int x=Search(temp,temp->data);
    printf("%s,%d \t",temp->data,x);
    DeleteKey(&temp,temp->data);
    temp=temp->next;
  }
}
```

Outputs

Figure 1: Q1 to Q4

Figure 2: Q5 to Q6

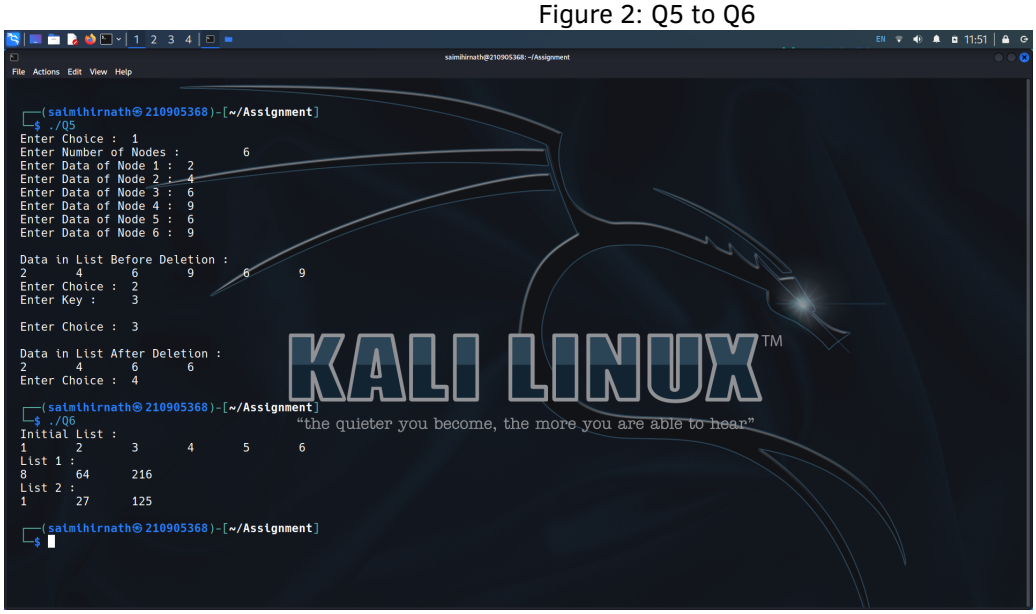

Figure 3: Q7 to Q10