# Assignment

Mushraf
AP19110010060
CSE-G.

1.

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node {
int value;
struct node *next;
};
void insert();
void display();
void delete();
int count();
typedef struct node DATA_NODE;
DATA_NODE *head_node, *first_node, *temp_node=0, *prev_node,
                                        next_node;
int data;
int main(){
int option=0;
Print("Singly linked list example -all options\n");
while(option<5){
Print("\n options\n");
print("1: Insert into linked list\n");
print("2: Delete from linked list\n");
Print("3: Display linked list\n");
Print("4: count linked list\n");
Print("others: Exit()\n")
Print("enter your option:");
scanf("%d", &option);
switch(option){
```

```
case 1:
  insert();
 break;
    case 2;
     delete();
     break;
   case 3:
     display():
     break;
   case 4;
   count();
   break;
    default;
     break;
   }
  }
  return 0;
}
void insert(){
Print("In Enter element of insert linked list :\n");
Scan("%d",& data);
temp_node=(DATA_NODE *) malloc (SIZE of (DATA_NODE));
temp_node->value=data;
if(first_node==0){
 first_node=temp_node;
}else{
 head_node->next=temp_node;
}
 temp_node->next=0;
```

```
Prev_node = temp_node;
temp_node = temp_node->next;
    }
  }
 }
}else
 Print("\n Invalid position\n\n");
}
 void display(){
 Int count = 0;
 temp_node = first_node;
 Printf("\n display linked list: \n");
 while (temp_node != 0){
 Print("# %d# ", temp_node->value);
 count++;
  temp_node = temp_node->next;
 }Print("\n No of terms in linked list: %d\n", count);
  }
  int count(){
  Int count = 0;
  temp_node = first_node;
  while (temp_node != 0){
   count++;
   temp_node = temp_node->next;
  }
  Print("\n no of terms in linked list: %d\n", count);
  return count;
  }
```

```
head_node=temp_node;
fflush(stdin);
}
void delete(){
int contvalue, pos, i=0;
contvalue = count();
temp_node = first_node;
printf("\n display linked list :\n");
Printf("\n Enter position for Delete element :\n");
Scanf("%d", &pos);
if(pos>0 && pos<=contvalue){
if(pos==1){
temp_node=temp_node->next;
first_node=temp_node;
Print("\n deleted successfully \n\n");
}else{
while(temp_node!=0){
if(i==(pos-1)){
Prev_node->next=temp_node->next;
if(i==(countvalue-1))
{
        head_node=prev_node;
    }
Printf("\n deleted successfully \n\n");
break;
}else{
i++;
```

2.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node* next;
};
void printlist(struct node * ptr = head;
while(ptr)
{
   Printf("%d->", ptr->data);
   ptr = ptr->next;
}
 Print("Null \n");
}
void push (struct node ** head, in data )
{
   struct node* newnode =(struct node *)malloc(sizeof(struct node));
   newNode->data= data;
   newnode->next= *head;
   *head = newnode;
}
struct node* Shuffle Merge (struct node *a, struct node* b)
{
   struct node dummy;
   struct node * tail = &dummy;
   dummy.next= NULL;
   while(1)
  {
   if(a==NULL)
   { tail->next=b;
     break;
   }
```

```
else
{
  tail->next=a;
  tail=a;
  a=a->next;
  tail->next=b;
  tail=b;
  b=b->next;
  }
 }
 return dummy.next;
}
int main(void)
{
 int keys[]={1,2,3,4,5,6,7};
 int n=sizeof(keys)/sizeof(keys[0]);
  struct node *a=NULL,*b=NULL;
  for(int i=n-1;i>=0;i=i-2)
        Push(&a,keys[i]);
  Print("First list:");
  PrintList(a)
   Print("Second list:");
   PrintList(b);
   struct node *head=ShuffleMerge(a,b);
   Print("After Merge:");
   PrintList(head);
   return 0;
  }.
```

3.

```
#include <stdio.h>
int top= -1;
int x;
char stack[100];
void push (int x);
char pop();
int main()
{
int i,n,a,t,k,f,som=0,count=1;
Print("enter the number of elements in the stack");
 scanf ("%d", &n);
 for (i =0; i<n; i++){
Printf ("enter next element");
Scanf ("%d",&a);
Push (a);
}
Printf ("enter the sum to be checked");
Scanf ("%d",&k);
for (i=0 ; i<n; i++)
{
 t= POP();
 som+=t ;
 count+=1;
 if (som==k){
 for (int j=0; j<count ; j++)
 Printf ("%d ", stack[j]);
 f=1;
 break;
 }
 push (t);
 }
```

```
if (f!=1)
Printf ("the elements in the stack dont add up to the sum");
}
void push (int x)
{
if (top==99)
{
Printf("\n stack is FULL!!:\n");
return;
}
top=top+1;
stack[top]=x;
}
char pop()
{
if (stack[top]==-1)
{
Printf("\n stack is EMPTY!!:\n");
return0;
}
x=stack[top];
top=top-1
return x;
}.
```

4.
```
#include<stdio.h>
# define SIZE 10
void insert(int);
void delete();
int queue[10], f=-1, r=-1;
void main(){
Int value, choice;
while(1){
Print("\n\n*** MENU ***\n");
```

```
Printf("1.Insertion\n2.Deletion\n3.Print Reverse\n4.Print Alternate\n5.Exit");
Print("\nEnter your choice:");
Scanf("%d",&choice);
switch(choice);
switch(choice){
case1: print["enter the value to be insert");
Scanf("%d",&value);
Insert(value);
break;
case 2:delete();
break;
case 3:
      Print("The reversed queue is:");
      for(int i=SIZE;i>=0;i--)
{
   if(queue[i]==0)
  continue:
     printf("%d",queue[i]);
}
     break;
  case 4:
     Print("Alternate elements of the queue are:");
     for(int i=0;i<SIZE;i+=2)
{
     if(queue[i]==0)
     continue:
     Print("%d",queue[i]);
}
     break;
  case5;exit(0):
   default printf("\n wrong selection!!! try again!!!");
   else {
       if(f==-1)
```

```
f =0;
    r=(r+1)%SIZE;
    queue[r]=value;
    Print("\nInsertion success!!!");
}}
void delete(){
 if (f==-1)
   Print("\nQueue is Empty!!!. Deletion is not possible!!!");
 else{
  Print("\nDeleted :%d", queue[f]);
  f=(f+1)%SIZE;
  if(f==r)
   f=r=-1;
  }}.
```

5(i) Difference between Array and Linked linked list the major difference between Array and linked list regards to their structure. Arrays are index based data structure where each element associated with an index. On the other hand, linked list relies on references where each node consists of the data and the references to the previous and next element.

(ii)
```
#include<stdio.h>
#include<stdlib.h>
struct node
 {
  int data;
  struct node * next;
 } *start1, *start2;
 void createlist (struct node start, int n);
 void main()
 {
```

```
struct node *temp1, *temp2;
int n, m, x
Pf("enter number of elements in list1");
sf ("%d", &n);
create list (start1, n)
Pf("enter number of elements in list2");
sf ("%d", &m);
create list (start2, m);
temp1 = start1;
temp2 = start2;
start2 = start2 → next;
temp2 → next = temp1;
start1 = temp2;
temp1 = start1;
temp2 = start2;
while (temp1 != NULL)
{ Pf("%d", temp1 → data);
temp1 = temp1 → next);
while (temp2 != NULL)
{
Pf ("%d", temp2 → data);
temp2 = temp2 → next;
}
}.
```