Assignment-4

B.V.P.S. Bhismarao
AP19110010319
CSE-F

(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 structnode DATA_NODE;
DATA-NODE* head_node, * first_node, * temp_node = 0, *prev_node,
                                                    next_node;

int data;
int main() {
    int option=0;
    printf("Singly Linked List\n");
    while (option < 5) {
    printf("\nOptions\n");
    Printf("1: Insert into linked list\n");
    printf("2: Delete from linked list \n");
    Printf("3: Display Linkedlist \n");
    Printf("4: Count linked list \n");
    Printf("0 thers: Exit() \n");
    Printf("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() {
    printf("\n Enter Element for linked list:\n");
    Scanf("%d", &data);
    temp_node = (DATA_NODE*) malloc (sizeof(DATA_NODE));
    temp_node -> value=data;
    if (first_node==0) {
      first_node= temp_node;
    }
    else {
    head_node -> next = temp_node;
   }
   temp_node -> next=0;
```

```
head_node = temp_node;
fflush(stdin);
}
voiddelete() {
    int countvalue, pos, i=0;
    countvalue= count();
    temp_node = first_node;
    printf("\n D'g playing Linked List: \n");
    printf("\n Enter position for delete element: \n");
    scanf("%d", &pos);
    if( pos>0 && pos<= countvalue){
        if (pos==1){
            temp_node =temp_node->next;
            first_node = temp_node;
            printf ("\n Deleted successfully \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("Deleted successfully\n")
        break;
        }
        else{
            i++;
```

```
        prev_node = temp_node;
        temp_node = temp_node->next;
      }
    }
  }
} else{
    printf("\n Invalid position \n");
}
void display(){
int count=0;
temp_node = first_node;
printf("\n Display linked list:\n");
 while (temp_node != 0) {
     printf("%d ", temp_node->value);
      count++;
      temp_node = temp_node->next;
 }
  printf("\n No of items in linkedlist: %d \n", count);
 }
  int count(){
     int count=0;
     temp_node = ~~temp_node->next~~ first_node;
      while(temp_node != 0){
          count++;
          temp_node = temp_node->next
       }
```

```
printf("\n No. of items in linkedlist: %d\n", count);
return count;
}
```

(2)

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
  int data;
  struct Node *next;
};
void printlist (struct Node* head)
{
   struct Node* ptr=head;
   while (ptr)
   {
     printf("%d->", ptr->data);
     ptr= ptr->next;
   }
   printf ("Null\n");
}
void push(struct Node **head, int data)
{
   struct Node* newNode= (struct Node*)malloc(sizeof (struct Node));
   newNode->data=data;
   newNode->next= *head;
   *head = newnode;
}
```

```
Struct Node* shuffleMerge (struct Node* a, struct Node* b)
{
    StructNode dummy;
    Struct Node *tail = &dummy;
    dummy.next = NULL;
    while(1)
    {
      if(a== NULL)
      {
         tail -> next= b;
          break;
       }
      elseif (b== NULL)
       {
         tail->next=a;
         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]);
    }
    for (int i = n-2; i >= 0; i = i-2) {
        push(&b, keys[i]);
    }
    printf(" First List:");
    printList(a);
    printf("second List");
    printList(b);
    struct Node *head = shuffleMerge(a, b);
    printf("After Merge");
    printlist(head);
    return 0;
}
```

3.
```
#include <stdio.h>
int top=-1;
int x;
Char stack[100];
voidpush(int x);
Char pop();
int main(){
    int i,n,a,t,K,f,sum=0, count=1;
    Printf(" Enter the numbers 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();
        Sumt+=t;
        Count+=1;
        if (sum==K){
            for(int j=0; j<count; j++)
            Printf("%d", stack[j]);
            f=1;
            break;
        }
        push(t);
    }
```

```
if(f1==1){
  printf("The elements in the stack donot adduptosum");
 }
}
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){
        printf("\n Menu \n");
        printf(" 1. Insertion \n 2. Deletion \n 3. Printreverse \n
                4. Print alternative \n 5. Exit");
        printf(" \n Enter your choice");
        scanf("%d", &choice);
        switch(choice) {
            Case 1: printf("Enter the value to be insert:");
                scanf("%d", &value);
                insert(value);
                break;
            Case 2:
                delete();
                break;
            Case 3:
                printf("The reversed queue is:");
                for(int i= SIZE; i>=0; i--)
                {
                    if (queue[i]==0)
                        continue;
                    Printf("%d", queue[i]);
                }
```

```
    break;
Case 4;
    printf("Alternate elements of the queue are:");
    for(int i=0; i<SIZE; i+=2)
    {
        if( queue[i]==0)
            continue;
            printf("%d", queue[i]);
    }
     break;
Case 5: exit(0);
default: printf("\n Wrong selection\n");
    }
}}
void insert(int value){
    if(( f==0 && r=SIZE-1) || f==r+1)
        printf("\n Queue is full insertion not possible");
    else{
     if( f==-1)
      f=0;
       r=(r+1)%SIZE;
       queue[r]=value;
       printf("\n Insertion success");
    }}
    void delete(){
     if(f==-1)
      printf("\n Queue is Empty deletion not possible");
     else{
```

```
printf("\n Deleted : %d", queue[f]);
f = (f+1) % SIZE;
if (f==r)
  f=r=-1;
}}
```

5

1) Difference between Array and 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.

2)
```
#include <stdio.h>
#include <stdlib.h>
Struct Node {
        int data;
        Struct Node* next;
};
void printList (struct Node* head)
{
  Struct Node* ptr = head;
   while (ptr)
   {
     printf ("%d->", ptr->data);
      ptr = ptr->next;
   }
```

```
    printf(" Null \n");
}
void push (struct Node** head, int data)
{
    struct Node* newNode = (struct Node*) malloc (sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}
void MoveNode (struct Node** destRef, struct Node** sourceRef)
{
    if (*sourceRef == NULL)
        return;
    struct Node* newNode = *sourceRef;
    *sourceRef = (*sourceRef)->next;
    newNode->next = *destRef;
    *destRef = newNode;
}
int main(void)
{
    int keys[] = {1, 2, 3, 4};
    int n = sizeof keys / sizeof (keys[0]);
    struct Node* a = NULL;
    for (int i = n-1; i >= 0; i--)
        push (&a, keys[i]);
```

```
struct Node* b= NULL;
for (int i=0; i<n; i++)
    push (&b, a[keys[i]);
 MoveNode(&a, &b);
 printf ("First List:");
 printList(a);
 printf(" Second List:");
 printList(b);
return 0;
}
```