Ch. Sripathi Vasavardhan
AP19110010410
CSE-F

# ASSIGNMENT-4

(1)

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.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;
 printf("Singly Linked List Example - All operations\n");
 while (option<5)
{
  printf("\n Options\n");
```

```
printf("1 : Insert into Linked List \n");
printf("2 : Delete from Linked List \n");
printf("3: Display Linked List \n");
printf("4: Count Linked List \n");
printf("Others : 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;
Case 5
default : break;
}
}
  return 0;
}
void insert()
{
printf("\n Enter Element for Insert 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);
 }
void delete()
{
  int countvalue, pos, i = 0;
  countvalue = count();
  temp_node = first_node;
  printf("\n Display Linked List: \n");
  print("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\n");
  } else
  {
  while(temp_node != 0)
  {
    if (i == (pos-1))
    {
      prev_node->next = temp_node->next;
      if (i == (countvalue-1))
      {
        head_node
      }
  printf("Deleted successfully\n\n");
  break;
  }
  else
  {
    i++;
  prev_node = temp_node;
  temp_node = temp_node->next;
  }
  }
} } else
```

```
printf ("\n Invalid position \n\n")
}
void display()
{
    int count = 0;
    temp_node = first_node;
    printf("Display Ranked List : \n");
    while(temp_node != 0)
    {
      printf ("# %d #", temp_node -> value);
      count ++;
      temp_node = temp_node -> next;
    }
  printf("\n No of Items in Ranked List : %d \n", count);
}
int count()
{
    int count = 0;
    temp_node = first_node;
    while (temp_node != 0)
    {
      count ++;
      temp_node = temp_node -> next;
    } printf ("\n No of Items in Linked List : %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)
{
   struct Node dummy;
   struct Node * tail = &dummy;
   dummy.next = NULL;
while(1)
{
    if(a == NULL)
    {  tail->next = b;
       break;
    }
    else if(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]);
    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];
void push(int x);
char pop();
int main()
{
 int i,n,a,t,k,f,sum=0,count=1;
 printf("Enter no. of elements in stack");
 scanf("%d",&n)
 for(i=0;i<n;i++)
 {
 printf("Enter next element");
 scanf("%d",&a);
 push(a);
 }
 printf("Enter sum to be checked");
 scanf("%d",&k);
 for(i=0;i<n;i++)
 {
    t=pop();
    sum+=t;
    count+=1
```

```
if (sum == k)
{
  for(int j=0;j<count;j++)

  printf("%d", stack[j]);
  f = 1;
  break;
 }
 push(t);
 }
 if (f != 1)
printf("the elements in stack donot 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");
```

```
    return 0;
}
    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\n *** MENU *** \n");
   printf (" 1. Insertion\n2. Deletion\n3. Print Reverse \n
            4. Print Alternatives  5. EXIT ");
   print ("\n enter your choice\n");
   scanf ("%d", &value);
   insert (value);
   insert (choice);
```

```
case 1 : printf("enter value to be inserted:");
         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("Alternative 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("In Wrong Selection!!! Try again!!!");
    }
```

```
}}
  }

Void delete()
{
  if (f == -1)
    printf("\n Queue is Empty !!! Deletion is not possible!!!");
  else {
    printf("\n Deleted : %d", queue[f]);
     f = (f+1) % SIZE
     if (f == r)
   f = f = r = -1;
  }
}
```

(b)

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

(5)

(a)

```
#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));
  new Node -> 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};
    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, 2*keys[i]);
    MoveNode(&a, &b);
    print("First List\n"); printList(a);
    print("second List\n"); printList(b);
    return 0;
}
```