M. Mohana Surendar
AP19110010467
Sec - CSE - F

1) Program to insert and delete an element at nth and Kth position in linked list

```
#include <stdio.h>
# include <stdlib.h>
struct linked_list
{
    int num;
    struct linked list *next;
};
struct linkedlist node;
 node *head=null, *last = Null;
void create _ linked_ list();
Void Display();
void insertatlast(int);
void insert atfirst (int);
void insertafter (int key, int value);
void delete (int);
void search (int);
int main().
{
  int key, value;
  printf( "create linked list \n");
```

```
createlinkedlist();
Display();
void insertatlast(int);
void insertatfirst(int);
void insertafter(int key, int value);
void delete(int value);
void search(int value);
//Insert value at last position
print("\n To insert new item at last");
printf
scanf("%d", &value);
insertatlast(value);
Display();
printf("Insert new Item at first\n");
Scanf("%d", &value);
insertatfirst(value);
Display();
printf("\n(Enter a key (Existing Item of list), and
    Enter value to insert\n");
scanf("%d", &key);
printf("\n Insert new Item after %d KEY\n", key);
Scanf("%d", &value);
insertafter(key, value);
Display();
```

```
printf("To search an element in linked list\n");
printf("\n Enter an element to search if found in
        list\n");
scanf("%d", &value);
searchitem(value);
printf("\n To delete an element in linked list\n");
printf("\n Enter value \n");
scanf("%d", value);
deleteitem(value);
  Display();
return 0;
}
void createlinkedlist()
{
  int val;
  while(1)
  {
    printf("Input a number, Enter -1 to exit \n");
    scanf("%d", &value);
    if(val == -1)
      break;
    insertatlast(val);
  }
}
```

```
void insut at last (int value)
{
    node *tempnode;
    tempnode = (node*) malloc(sizeof(node));
    tempnode -> num  =value;
    tempnode -> next = Null;
    if(head = Null)
    {
        head = tempnode;
        last = tempnode;
    }
    else
    {
        last -> next =tempnode;
        last = tempnode;
    }
}
Void insut at first (int value).
{
    node * tempnode =(node*) malloc(sizeof(node));
    tempnode -> numbu =value;
    tempnode -> next = head;
    head =tempnode;
}
void insertafter ( int key, int value)
{
    node *mynode = head;
    int flag =0;
```

```
while (mynode != Null)
{
  if(mynode ->number == key)
  {
    node *newnode = (node*) malloc(sizeof (node));
    newnode->number = value;
    newnode->next = mynode->next;
    mynode->next = newnode;
    printf("%d is inserted after %d \n", value, key);
    flag = 1;
    break;
  }
  else
    mynode = mynode->next;
}
if (flag == 0)
      printf("key not found \n");
}
void deleteitem(int value)
{
  node *mynode = head; *previous = Null;
  int flag = 0;
  while (mynode != Null)
```

```
{
  if(mynode -> number == value)
  {
    if(previous == Null)
      head = mynode -> next;
    else
      previous -> next = mynode -> next;
    printf("%d is deleted from list\n", value);
    flag=1
    free(mynode);
    break;
  }
  previous = mynode;
  mynode = mynode -> next;
}
if(flag==0)
    printf("Key not found!!\n");
}
void display()
{
  printf("\n your full linked list is\n");
  node *mylist;
  mylist = head;
  while(mylist != Null)
```

```
{ printf("%d ", mylist -> number);
  mylist = mylist -> next;
}
puts("");
}
```

Output

```
Create linked list
Input a number (Enter to exit)
1
Input a number ( " " )
2
Input a number ( " " )
3
Input a number ( " " )
4
Input a number ( " " )
5
Input a number ( " " )
1
Your linked list is
1 2 3 4 5
Insert new Item at last : 6
1 2 3 4 5 6
```

Insert new item at first

0

your linked list is

0123456

Enter a key (existing item of list), after that you want to insert a value.

6

Insert new item after 6 key

7

7 is inserted after 6.

your full linked list is

01234567

Enter an item to search it from list

3

3 is present in list. memory address is

296847686

Enter a value, which you want to delete from list

5

5 is deleted from list

your full linked list is

0123467

2) construct a new linked list by merging alternate nodes of two lists

```
#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 * Shuffle Merge( Struct Node *a, Struct
                             Node* b)
{
   Struct Node Alex;
   Struct Node *tail = &Alex;
   Alex.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 Alex.next;
}
```

```
{
    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;
}
```

Output

First list : 1 → 3 → 5 → 7 → null

Second list : 2 → 4 → 6 → Null.

After merge : 1 → 2 → 3 → 4 → 5 → 6 → 7 → null

3)

```
#include <stdio.h>
int stack[100], choice, n, top, x, i;
void push (void);
void display (void);
int main()
{
  top = -1;
  printf("\n Enter the size of Stack: ");
  scanf("%d", &n);
  printf("\n\t STACK OPERATIONS USING ARRAY");
  printf("\n\t 1. PUSH\n\t 2. DISPLAY\n\t 3) SUBARRAY
                \n\t 4. EXIT");
  do
  {
    printf("\n Enter the choice: ");
    scanf("%d", &choice);
    switch (choice)
    {
      case 1:
      {
        push()
        break;
      }
      case 2:
      {
        display()
        break;
      }
```

```
case 3:
{
  Sub Arraysum()
  break;
}
case 4:
{
  printf("\n\t EXIT POINT ");
    break;
}
default:
{
  printf("\n\t Pheare Enter a Void choice
                    (1/2/3/4)");
}
}
}
while(choice != 4)
return 0;
}
Void  Push()
{
  if (top >= n-1)
    {
      printf( "\n\t STACK is overflow");
    }
```

```
else
{
    printf("Enter a value to be pushed");
    scanf("%d", &x);
    top++
    stack[top]=x;
  }
}
void display()
{
  if(top>=0)
  {
    printf("\n The elements in stack\n");
    for(i=top; i>=0; i--)
      printf("\n %d", stack[i]);
    printf("\n press Next choice");
  }
  else
  {
    printf("\n The stack is empty");
  }
}
```

```
int subArray Sum (int stack[], int sum);
{
    int curr_sum, i, s;
    scanf("%d", &sum);
    for(i=0; i<n; i++)
    {
        curr_sum = stack[i] + stack[j];
        for [j = i+1; j<=n; j++)
        {
            if (curr_sum == sum)
            {
                printf("sum found %d and %d, stack[i],
                                    stack[j];
                return 1;
            }
            if (curr_sum > sum || j == n)
                break;
            curr_sum = curr_sum + stack[j];
        }
    }
    printf("No subarray found");
    return 0;
}
```

```
int main()
{ int sum = 23;
  SubArraySum (stack, n, sum);
  return 0;
}
```

Output

```
1. PUSH
2. DISPLAY
3. SUBARRAY
4. EXIT.

Enter choice 1
Enter value to be pushed
1
Enter choice : 1
2
Enter choice : 1
3
Enter choice : 1
4
Enter choice : 2
The elements in stack are
1
2
3
4
```

Enter choice 3

3

Sum found 1, 2

4)

```
#include <stdio.h>
#define size 20
void insert(int);
void delete()
int queue[20], a=-1, b=-1;
void main()
{
  int num, choice;
  while(1)
  {
    printf("1. Insert 2 delete \n 3 print
    \n 4 Reverse \n 5 Alternate \n 6 Exit)
    printf("\n Enter your choice");
  Scanf("%d", &choice);
  Switch (choice)
case 1: printf("Enter the num to insert:");
  Scanf("%d"; &num);
  insert(num);
  break;
```

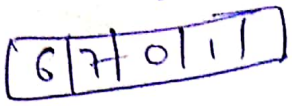
```
case 2 : printf( "Reverse queue");
    for (int i=size, i>0, i--)
    if (queue[i]==0)
      continue;
    printf("%d", queue[i]);
  }
  break;
case 3:
  printf( "Alternate elements");
    for(int i=0, i<size, i>0, i=i+2)
  {
    if(queue [i]==0)
        continue;
      printf( "%d", queue[i]);
  }
  break;
  return 0;
}
```

5) Array vs Linked lists.

1) Both are the data structure, Both are used to store data.

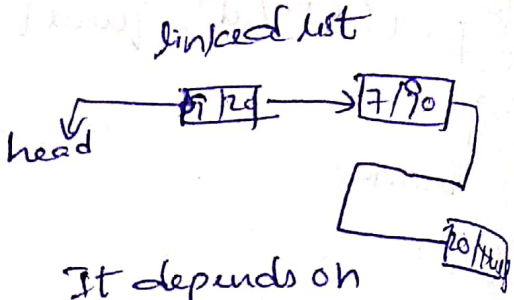
2) cost of accessing the elements

Arrays



* It takes as constant time O(1)

linked list



It depends on number of nodes in linked list O(n)

3) Memory requirement and utilization

Array

* Ineffective in memory utilization

ex

8x4=32 bytes

used =12

* Required memory is less in less data

linked list

* It is dynamic byte

8x3=24 byte

* Require more memory

4) Time of Insertion and time complexity of deletion.

Array

Beginning O(n)

At end O(1)

In position O(n)

linked list

O(1)

O(n)

O(n)

5) Easy use and operation

Array

* easy to use
→ linear and binary

Linked list

* less easier
→ linear

ii)

```
#include <stdio.h>
#include <stdlib.h>
int len(int a[])
{ int i=0,x,y=0
  while(1)
  {
    if(x[i])
    {
      xy++,i++;
    }
    else
    {
      break;
    }
  }
  return xy;
}
void changelist(int x[],int a[])
  {
    for(int i = len(x)-1; i>=0;i--)
    {
      x[i+1]=x[i]
    }
```

```
x[0]=a[0];
printf("\nElements of old array:\n");
for(int i=0; i<len(x); i++)
  {printf("%d", x[i])
  }
for(int i=0; i<len(y); i++)
  {
   y[i]=y[i+1];
  }
printf("\n Elements of new array:\n")
for(int i=0; i<len(a); i++)
{printf("%d", a[i]);
}
int main()
{
 int x[10]={1,2,3}, a[10]={4,5,6};
 change(a,b);
}
```