1) Program to insert and delete an element at nth & kth position in linked list.

T. Durgaprasad
AP19110010212
CSE-G

```
#include<stdio.h>
#include<stdio.h>
struct linked-list
{
   int number;
    struct linked-list* next;
};
typed of struct linked-list node;
node *head=NULL,*last=NULL;
void create_linked_list();
void print_linked_list();
void insert_at_last(int value);
void insert_at_first(int value);
void insert_after(int key, int value);
void delete_item(int value);
void search_item(int value);
int main()
{
  int key, value;
//Create_linked_list();
  Print_linked_list();
```

```
// Insert value at last position to existing linked list.
    Printf("\n Insert new item at last\n");
    Scanf("%d", &value);
    insert_at_last(value);
    Print_linked_list();
// Insert value at First position to existing linked list.
    Printf("\n Insert new item at first\n");
    Scanf("%d", &value);
    insert_at_first(value);
    Print_linked_list();
// Insert value after a defined value
    Printf("\n Enter a KEY (existing item of list),
                after that you want to insert a value\n");
    Scanf("%d", &Key);
    Printf("\n Insert new item after %d KEY\n", Key);
    Scanf("%d", &value);
    insert_after(Key, value);
    Print_linked_list();
```

```
// Search an item from linked list.
printf("\n Enter an item to search it form list \n");
scanf("%d", &value);
search_item(value);

// Delete value from linked list
printf("\n Enter a value, which you want to delete \n");
scanf("%d", value);
delete_item(value);
print_linked_list();

return 0;
}

/*
User defined functions
*/
```

```
Void create_linked_list()
{
  int val;
  while()
{
  Printf("Input a number. (Enter -1 to Exit) \n");
  Scanf("%d", &value);
  if (val == -1)
      break;
  insert_at_last(val);
  }
}
void insert_at_last(int value)
{
  node * temp_node;
    temp_node = (node*) malloc(sizeof(node));
  temp_node -> number = value;
  temp_node -> next = NULL;
 // For the 1st element
   if(head = NULL)
  {
    head = temp_node;
    last = temp_node;
  }
```

```
else
 {
  last->next=temp_node;
  last = temp_node;
  }

}
void insert_at_first(int value)
{
 node *temp_node = (node*) malloc (sizeof(node));
 temp_node->number= value;
 temp_node->next= head;
  head = temp_node;
}
 void insert_after(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 = myNod -> next;

  }
  if (flag == 0)

      Printf ("Key not found! \n");

}
```

```
void delete_item(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 Print_linked_list()
{
  Printf("In your full linked list is\n");
    node *mylist;
    mylist = head;
    While(mylist != NULL)
  {
    Printf("%d", mylist->number);
     mylist = mylist-> next;
    }
     Puts("");
}
```

1st output:

Create Linked List

Input a number. (Enter -1 to exit)

1

Input a number. (Enter -1 to exit)

2

Input a number. (Enter -1 to exit)

3

Input a number. (Enter -1 to exit)

4

Input a number. (Enter -1 to exit)

5

Input a number. (Enter -1 to exit)

-1

Your full linked list is

1 2 3 4 5

Insert new item at last

1 2 3 4 5 6

Insert new item at first

0

Insert Your full linked is

0 1 2 3 4 5 6

Enter a Key (existing item of List), after that you want to insert a value

6

Insert a new item after b Key

7

7 is inserted after 6

Your full linked list is

0 1 2 3 4 5 6 7

Enter an item to search it from List

3

3 is present in list. Memory address is 12348688

Enter a value, which you want to delete from list

5

5 is deleted from list

Your full linked list is

0 1 2 3 4 6 7

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

```
#include <stdio.h>
#include <stdlib.h>
// Data structure to store a linked list
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");
// Insert new node in begining
void Push (struct Node** head, int data)
{
  struct Node* newNode = (struct Node*)
      newNode->data = data;
      newNode->next = head;
   *head = newNode;
}
```

// Function to construct a linked list by merging alternate nodes // two given linked lists using dilbar node

```
Struct Node* ShuffleMerge(Struct Node* a, Struct Node* b)
{
   Struct Node dilbar;
   Struct Node* tail= &dilbar;
    dilbar.next= NULL;
    while(1)
    {
      // empty list cases
       if(a== NULL)
        tail->next= b;
         break;
       }
       else if (b==NULL)
       {
        tail->next= a;
          break;
       }
       else if(b==NULL)
       {
        tail-> next= a;
         break;
       }
      // move two nodes to tail
      else
        {
          tail->next=a;
          tail = a;
```

```
        a = a->next;
      tail->next = b;
      tail = b;
    b = b->next;
  }
  }
  return dilbar.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;
}
```

Output:

First List: 1 → 3 → 5 → 7 → null

Second List: 2 → 4 → 6 → null

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

2)

```
#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:
{ SubArray somC)
    break;
 }
case 4:
  { Printf ("\n\t EXIT POINT");
   break;
  }
  default:
  {
   Printf ("\n\t Please Enter a valid 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 SubArraySum(int stack[], int sum):
  {
   int curr sum, i, J;
    Scanf("%d", &Sum);
    for(i=0; i<n; i++)
    {
     curr-sum=Stack[i]; Stack[J];
```

```
//try all Subarrays starting with;
  for(J=i+1; J<=n; J++)
  {
    if(Curr-Sum==Sum)
    {
      Printf("Sum found %d and %d,
        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;
     Sub Array Sum(Stack, n, Sum);
      return 0;
  }
```

Output:-

3rd Output:

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

Enter choice: 1
Enter a value to be Pushed:
1
Enter choice: 1
Enter a value to be Pushed:
2
Enter choice: 1
Enter a value to be Pushed:
3
Enter choice: 1
Enter a value to be Pushed:
4
Enter choice: 2
The elements in stack.
1
2
3
4
Press next choice: 3
3
Sum found: 1, 2

Q)

(1) implement of quee in Reverse Order

I used stack to Reverse quee.

```
#include <stdio.h>
#include<stdio.h>
# define MAX 20
void show (int stack [], int size, int top)
{
int i;
for(i=0; i<size; i++)
{
Printf ("\n Value at %d is %d.", top, stack(top));
  top = top - 1;
}
}
Void reverse (int stack [], int qu[], int *t, int *r,
                                              int *f)
{
 *f = 0;
 while(*t>-1)
 {
  *r = *r+1;
  qu[*r] = stack [*t];
  * t = * t-1;
 }while (*f <= *r)
  {* t = *t+1;
   Stack [*t] = qu[*f];
```

```
        *f=*f+1;
    }
}
int main()
{
 int size
 int item, t, i, stack[MAX], quee[MAX];
 int top=-1, front=-1, rear=-1;
 Printf("Enter size of stack");
 Scanf("%d", &size);
 for(i=0; i<size; i++)
 {
  top=top+1;
  Printf("Enter value of for Postion %d : : ", top);
  Scanf("%d", &item);
  Stack[top]=item;
 }
 Show(stack, Size, top);
 reverse(stack, quee, &top, &rear, &front);
 Printf("\n After Reverse ---");
 Show(stack, size, top);
  getch();
}
```

Output:

Enter size of stack:5

Enter value of forpostion 0::1

Enter value of for position 1::2

" " " " " 2::3

" " " " " 3::4

" " " " " 4::5

value at 4 is5

value at 3 is4

value at 2 is3

value at 1 is2

value at 0 is 7

After reverse...

value at 4 is 1

Value at 3 is 2

Value at 2 is 3

Value at 1 is 4

value at 0 is 5

9 (ii) Program to print elements in a quee in alternate order

```
#include <stdio.h>
#define MAX 50
 void insert ();
 void alternate ();
 void display ();
 int queue_array[MAX];
  int rear=-1;
 int front=-1; size;
→ Scanf ("%d", & Size);
   main()
 { int choice;
    while()
   { Printf ("1. Insert element to quee \n");
     Printf ("2. Display element from quee \n");
     Printf ("3. Alternate elements");
    Printf ("4. Quit \n");
     Printf ("Enter your choice:");
      Scanf ("%d", & choice:");
      Switch (choice)
       {Case 1;
        insert ();
         break;
```

```
case2:
display();
 break;
case3:
alternate();
 break;
 case4:
 exit(1);
 default:
    Printf("Wrong choice\n");
  }
 }
}
voidinsert()
{
 int add-item;
 if(rear == MAX-1)
 Printf("Reverse Overflow\n");
 else
 {
  if(Front==-1)
   front = 0;
  Printf("Insert the element in Quee");
  Scanf("%d", &add-item);
   rear = rear+1;
   qvee-array[rear]= add_item;
   }
}
```

```
void display ()
{
  int i;
    if (front == -1)
     Printf("Queee is empty \n");
    else
    { printf (" Queee is: \n");
    for(i=front; i<= rear; i++)
        Printf("%d", quee-array[i]);
        Printf ("\n");
       }
}
voidalternate()
{
  int i, J, temp;
  Printf("alternate elements are \n");
  for(i=0; i<size; i+=2)
    Printf("%d \n", quee-array[i]);
}
```

Output:

1. Enter choice: 1
Insert the element in quee: 10

Enter choice: 1
Insert the element in quee: 20

Enter choice: 1
Insert the element in quee: 30

Enter choice: 1
Insert the element in quee: 40

Enter choice: 1
Insert the element in quee: 50

Enter choice: 2
10
20
30
40
50

Enter choice: 3
10
30
50

Enter choice: 4

Exit

5 ii) How array is different from linked list?

Array	Linked list
1) An array is a collation of elements of a similar datatype	1) Linked lists is an ordered collation of elements of same type in which each element is connected to next using pointers.
2) Array elements can be accessed randomly using the array index	2) Random accessing is not possible in linked lists The elements will have to be accessed sequentially.
3) Data elements are stored in contigous locations in memory	3) New elements can be stored anywhere and a refrence is created for the new element using pointers

5 (iii) Program to add first node of linked list to another linked list:

```
#include <stdio.h>
#include<stdlib.h>
    Struct Node
    {
      int data;
      Struct Node* next;
    };
    void Print List (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;
  }
// Function take the node from the front of source
// and move it to front of destination
void Move Node(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]);
  // construct 1st linked list
  // construct 2nd linked list
  struct Node *b = NULL;
  for(int i=0; i<n; i++)
    push(&b, 2*keys[i]);
  //move front node of b and move it to the front of a
  moveNode(&a, &b);
  printf("First List:");
  printList(a);
  printf("Second List:");
  printList(b);
```

return 0;

}

Output:

First List: 0→1→2→3→null

Second List: 4→2→null