1) Program to insert and delete an element at n^{th} & K^{th} position in linked list.

K. Padma Lokesh
AP19110010021
CSE-G

```
#include<stdio.h>
#include<stdlib.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 Prit_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 a linked list
printf("create linked list\n");
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 poistion to existing linked list.
printf("\n Insert new item at first \n");
Scanf("%d", &value);
insert_at_first(value);
Prine_linked_list();
// Insert value after a defined value
printf("\n Enter aKEy(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 at item from linked list.
    printf("\n Enter an item to search it from 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 crete_linked_list()
{
```

```
int val;
while(1)
{
  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 (size of (node));
temp_node -> number = value;
temp_node -> next = head;
head = temp_node;
}
void insert_after (int key, int value)
{
node * my node = head;
int flag = 0;
while (my Node != NULL)
{
if (my Node -> number == key)
{
node * newNode = (node*) malloc (size of (node));
new Node -> number = value;
new Node -> next = my Node -> 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)
```

```
    {
    last -> next = temp_node;
    last = temp_node;
    }
}
void insert_at_first (int value)
{
node * temp_node = (node*) malloc (size of (node));
temp_node -> number = value;
temp_node -> next = head;
head = temp_node;
}
void insert_after (int key, int value)
{
node * my node = head;
int flag = 0;
while (my Node != NULL)
{
if (my Node -> number == key)
{
node * newNode = (node*) malloc (size of (node));
new Node -> number = value;
new Node -> next = my Node -> next;
```

```
        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("\n Your full linked list is \n");
 node * myList;
 myList = head;
 while (myList != NULL)
```

5 is deleted from list

```
    {
      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

Your full linked list 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 new item after 6 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 1234688

Enter a value, which want to delete

5

5 is deleted from list

0 1 2 3 4 6 7

2) Construct 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 newnode in begining
void Push (struct Node ** head, int data)
{
        struct Node * newNode = (struct Node*) malloc
                                Size of (struct No
        newNode → data = data;
        newNode → next = *head;
        *head = newNode;
}
// Function to construct a linked list by mergin
alternate node
// two given linked lists using dilbar node
Struct Node* Shuffle Merge (struct Node * a, struct
{
    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;
 }
 // move two nodes to tail
 else
 {
    tail->next=a;
    tail=a;
    a=a->next;
    tail->next=b;
    tail=b;
    b=b->next;
}
```

```
    }
    return dibar.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("\nAfter 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");
    Print f("\n\t 1. PUSH \n\t 2. DISPLAY\n\t 3. SUBAR
                                AY\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 Array sum()
    break;
  }
  case 4:
  { printf("\n\t EXIT POINT");
    break;
  }
```

```
        default:
        {
        printf("\n It Please Enter a vdid choice
                                    (1|2|3|4)");

        }

    }
}
while (choice != 4)
return 0;
}
void push()
{
  if (top>=n-1)
  {
   printf("\n It STACK is over flow");
  }
  else
  {
    printf("Enter a value to be Pushed");
    scanf("%d", &x);
     top++
     stack (top):x;
  }
```

```
Void display()
{
  if (top >=0)
   {
    Printf ("In the element 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 Sub Array sum (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 i
```

```
for (j=i+1; j<=n; j++)
{
  if (curr_sum == sum)
  {
    printf("sum found %d and %d", i, 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 = 13;
  subArraysum(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

}

sum found 1, 2

4) (i) implement of queue in Reverse order

⇒I used stack to Reverse queue

```
# include < conio.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 % ;
}
while (*f <= *r)
{
*t = *t + 1;
stack[*t] = qu[*f];
*f = *f + 1;
}
}
int main()
{
int size
int item, t, i; stack[MAX], que[MAX];
int top = -1, front = -1, rear = -1;
printf("Enter size of stock);
Scanf("%d", &size);
for (i = 0; i < size; i++)
{
```

```
        top = top+1;
        printf("Enter value of for position %d ::",top);
        scanf("%d", &item);
        Stack[top]=iteam;
    }
    show(stack,size,top);
    reverse(stack, queue, &top, &rear, &front);
    printf("\n After Reverse.....");
    show(stack,size,top);
    getch();
}
```

Output:

```
Enter size of stack : 5
Enter value of for position 0 : : 1
"  "  "  "  "  1 : : 2
"  "  "  "  "  2 : : 3
"  "  "  "  "  3 : : 4
"  "  "  "  "  4 : 5
Value at 4 is 5
```

Value at 3 is 4

" " 2 " 3

" " 1 " 2

" " 0 " 1

After reverse ---

Value at 4, is 1

" " 3 " 2

" " 2 " 3

" " 1 " 4

" " 0 " 5

4) iii) Program to print elements in a queue in alternate order

```
# include <stdio.h>
# include 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(1)
  {
    printf("1. Insert element to queue \n");
    printf("2. Display element from queue \n");
    printf("3. Alternate elements");
    printf("4. Quit \n");
    printf("Enter your choice:");
    scanf("%d", &choice);
    switch(choice)
    {
      case 1;
      insert();
      break;
      case 2:
      display();
      break;
```

```
        case 3:
        alternate();
        break;
        case 4:
        exit(1);
        default:
        printf("wrong choice\n");
      }
    }
}
void insert()
{
  int add_item;
  if (rear==MAX-1)
  Printf ("Queue overflow\n");
  else
  {
    if (front==-1)
    front:0;
    printf("Insert the element in Queue");
    scanf("%d", &add_item);
    rear:rear+1;
```

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

}

Output:

```
Enter choice:1
insert the element in queue: 10
Enter choice: 1
Insert the element in queue: 20
Enter choice: 1
Insert the element in queue: 30
Enter choice: 1
Insert the element in queue: 40
Enter choice: 1
Insert the element in queue: 50
Enter choice: 2
 10
 20
 30
 40
 50
Enter choice: 3
 10
 30
 50
```

Enter choice: 4

Exit

5) i) How array is different from linked list?

Array	Linked List
1) An array is a collection of elements of a similar data type	1) Linked lists is an ordered collection of elements of same type in which each element is collected 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 contiguous locations in memory	3) New elements can be stored anywhere and a reference is created for the new element using pointers

5) ii) 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 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
                                    (structNode));
 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 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(key[0]);
    struct Node* a = NULL;                 } // construct
    for (int i = n-1; i >= 0; i--)         } 1st linked list
(struct    push(&a, keys[i]);
node));
    // construct 2nd linked list
```

Scanned with CamScanner

```
Struct Node * b = NULL;
for (int i=0; i<n; i++)
    push (&, 2* keys[i]);
//move front node of b and move it to the fro
of a
moveNode(&a, &b);
Printf ("First List:");
printf ("second List:");
 Print List (b);
 return 0;
}
```

output:-

First List : b → 1 → 1 → 3 → null

Second List : 4 → 2 → null