MILIND GUPTA -AP19110010498

(1) Write a program to insert and delete an element at the nth and kth pointer in an linked list. where n and k are taken from the users.

```
# include <stdio.h>
# include <stdlib.h>
 struct node
 {
  struct node * next;
 };
  struct node * curr, * temp;
    Void Input (struct node *)
    Void delete (struct node *)
    Void main (void)

  {
   Struct node * s;
   int n;
   s = Null;
   do
   {
    Printf("Enter the element to insert: \n;");
    Printf(" 2. Delete \n");
    Print f(" 3. Exit \n");
    print f (" Enter the choice: ");
```

2020/4/20 21:32
REDMI NOTE 7
MILAN

```
Scanf("%d", &n);
        Switch (n)
        {
          Case 1: input (s);
                  break;
          Case 2: delete (s);
                  break ;
         } while (n: = 3)
   }
Void input (struct node * z)
 {
 int pos., c = 1
 curve z ;
 print f (" Enter the element to be inserted!");
  Scan f ("%d", & pos);
       while ( curv -> next b = Null)
       {
         c++;
         Pf (c == pos)
       {
        temp = (struct node *) malloc (size f
                                  (struct node));
        Point f ("Enter the numbers ; ");
        Scan f ("%d' & temp -> n);
```

REDMI NOTE 7
MILAN
2020/4/20 21:33

```
        temp -> next = curr -> next;
        curr -> next = temp;
        break;
      }
    }
  }
Void delete (struct node *z)
{
  int pos, c=1;
    curr = z;
    printf("Enter the element to be delete:");
    scanf("%d", &pos);
  while (curr -> next != NULL)
  {
    c++;
    if (c= pos)
  {
    temp = current -> next;
    curr => next = curr -> next -> next;
    free (temp)
    }
  curr = curr -> next;
```

REDMI NOTE 7
MILAN
2020/4/20 21:34

```
}
void merge (struct node *p, struct node *q)
{
 struct node *p_curr = p, *q_curr = *q;
 struct node *p_next, *q_next;
  while (p_curr != NULL && q_curr != NULL)
 {
   p_next = p_curr -> next;
   q_next = q_curr -> next;
   q_curr -> next = p_next;
   p_curr -> next = q_curr;
   p_curr = p_next;
   q_curr = q_next;

 }
 *q = q_curr
}
int main ()
 {
  struct node *p = NULL, *q = NULL;
    Push (&p, 1);
    Push (&p, 2);
    Push (&p, 3);
    Printf ("First linked list: \n");
    Print list (R);
```

REDMI NOTE 7
MILAN
2020/4/20 21:34

```
    Push (&q, 4);
    Push (&q, 5);
    Push (&q, 6);
    Printf(" second linked list:\n");
    Print list (q);
    merge (p, &q);
    Printf (" modified first linked list =\n");
    Print list (p);
    Printf (" modified second linked list = \n");
    Print list (q);

    return 0;
}
```

REDMI NOTE 7
MILAN
2020/4/20 21:34

2. Construct a new linked list by merging alternate nodes of two lists for example in list 1 we have {1,2,3} and in list 2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}.

```
#include <stdio.h>
#include <stdlib.h>
// Data structure to store a linked list
struct Node
{
  int data;
  struct Node* next;
};
// Helper function to print given linked list
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;
}
```

REDMI NOTE 7
MILAN
2020/4/20 21:36

```
// Function to construct linked list by merging
void merge(struct Node** a, struct Node** b)
{
 struct Node dummy;
 struct Node* tail = &dummy;
 dummy.next = NULL;
 while(1)
 {
   //empty list cases
   if (*a == NULL)
   {
     tail -> next = NULL;    // Note
     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;
   }
  }
  *a = dummy.next;
}
// main method
int main(void)
{
 struct Node *a = NULL, *b = NULL;
 //construct first list
 for(i=3; i>=1; i--)
   push(&a, i);
```

REDMI NOTE 7
MILAN
2020/4/20 21:35

```
// construct second list
for (int i = 6 ; i >= 4; i--)
  push(&b, i);

// print both linked list
printf("First List : ");
printList(a);
printf("Second list: ");
printList(b);
merge(&a, &b);
printf("\n After Merge :\n\n");
 printf("Final list :\n");
  print List(a);
  return 0;
}
```

Output

First List : 1→2→3

Second List : ~~3→4~~
4→5→6

After Merge:

Final list:

1→4→2→5→3→6

2020/4/20 21:36
REDMI NOTE 7
MILAN

3. Find all the elements in the stack whose sum is equal to k. (where k given from the user)

```
#include <stdio.h>
int x, top = -1;
int stack [100];
void push (int x)
{
if (top == 99)
{
printf (" stack is full OVERFLOW!!(");
}
top = top + 1,
stack [top] = x;
{
printf (" stack
char pop()
{
if (stack [top] == -1)
{
printf (" stack is empty UNDERTAKEN!!");
}
x = stack [top];
top = top -1;
return x;
}
void main()
{
int i, n, a, p, s, r, sum = 0, count = 1;
print_f ("Enter number of elements");
scanf ("%d", &n);
```

REDMI NOTE 7
MILAN
2020/4/20 21:36

```
for(i=0; i< n; i++)
{
 printf("Enter element");
 scanf("%d", &a);
 push(a);
}
printf("Enter the sum to be checked");
scanf("%d", &s);
for(i=0; i<n; i++)
{
 p=pop()
 sum += p
 count += 1
 if (sum == s)
 {
  for (int j=0 ; j<count , j++)
  printf("%d", stack[j]);
  r=1;
  break;
 }
 push (b);
} else
 { printf(" The elements in the stack don't add");
 }
```

Output:-

```
Enter number of elements: 4
Enter element 1
Enter element 2
Enter element 5
Enter element 3
Enter sum to be checked 8
1 2 5
```

2020/4/20 21:36
REDMI NOTE 7
MILAN

4. Write a program to print the elements in a queue.
(i) In reverse order
(ii) In alternate order

```
#include <stdio.h>
#define SIZE 10
int queue [SIZE], f= -1, r= -1

Void enqueue (int value)
{
if(cp==0 && r= SIZE-1 || f==r+1)
{
 print f(" OVERFLOW!");
}
 else
{
 if (f==-1)
 f=0;
 r= (r+1)% SIZE
  queue [r] = value;
 .printf (" Insertion successful");
}
}
 Void  dequeue()
 {
 if (f== -1)
 printf (" UNDERFLOW");
 else
{
printf {" Deleted element ", queue [f]);
f= (f+1)% SIZE
if(f==r)
```

REDMI NOTE 7
MI AN
2020/4/20 21:37

```
f= r=-1
}
}

void main()
{
int value, choice;
while(1)
{
printf("1. Insertion 2. Deletion 3. Print Reverse
        4. Print alternate 5. Exit");

Scanf("%d", &choice);
Switch (choice)
{
Case 1: printf("Enter the value to be inserted");
Scanf("%d", &value);
enqueue(value);
break;
Case 2: dequeue();
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;
```

REDMI NOTE 7
MILAN
2020/4/20 21:37

```
Case 4: printf ("Alternate element of the queue are:");
for (int i=0; i<SIZE; i=i+2)
{
if (queue [i] ==0)
continue;
printf("%d", queue[i]);
}
break;

Case 5: exit (0);
default : printf ("Invalid input");
}
}
}
```

OUTPUT -

1. Insertion 2. Deletion 3. Print ~~orde~~ reverse
4. print alternate 5. Exit

1

Enter the value to be inserted · 3

1

Enter the value to be inserted 5

3

5 3

4

3

REDMI NOTE 7
MILAN
2020/4/20 21:37

5(i) How array is different from the 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 to the previous and next element.

(ii) Write a program to add the first element of one list to a another list for example we have {1,2,3} in list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2

```
# include <stdio.h>
# include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
void push (struct node ** head_ref, int new_data)
{
    struct node *new_node = (struct node*) malloc (size of(
                                                  struct node));
    new_node => data = new_data;
    new_node => next = (*head_ref);
    (*head_ref) = new_node;
}
void printList (struct node *head)
{
    struct node *temp = head;
    while (temp != NULL)
{
```

REDMI NOTE 7
MILAN
2020/4/20 21:37

```
        printf ("%d", temp -> data);
        temp = temp -> next;
    }
    printf ("\n");
}
void merge (struct node *p , struct node **q)
{
    struct node *p_curr = p, *q_curr = *q;
    struct node *p_next , *q_next;
    while (p_curr != NULL && q_curr != NULL)
    {
        p_next = p_curr -> next;
        q_next = q_curr -> next;
        q_curr -> next = p_next;
        p_curr -> next = q_curr;
        p_curr = p_next;
        q_curr = q_next;
    }
    *q = q_curr;
}
int main()
{
    struct node *p = NULL , *q = NULL;
    push (&p, 1);
    push (&p, 2);
    push (&p, 3);
    printf ("First Linked List : \n");
    printList (p);
    push (&q, 4);
```

REDMI NOTE 7
MILAN
2020/4/20 21:37

```
push (&q,5);
push (&q,6);
printf (" second linked list : \n");
print list (q);
merge (p, &q);
Printf ("modified first linked list: \n");
Print List (p);
Printf ("modified second linked list : \n");
Print list (q);
getchar();
return 0;
}
```

Output:-

The elements of first array: {4, 1 2 3}

The element of second array 5 6

REDMI NOTE 7
MILAN
2020/4/20 21:38