Assignment-4

K. Nainekka

AP19110010184

Cse-G.

1. write a c program to insert and delete an elem at the nth and kth position in a linked list when n and k is taken from user

<u>Code:-</u>

```
# include <stdio.h>
# include <stdlib.h>
 struct Node
 {
     int data;
     struct Node * next;
  };
 void push(struct Node ** head_ref , int new_dat
 {
 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 delete node (structNode ** head_ref, int position
```

Scanned with CamScanner

```
{
  if(*head_ref==Null)
     return;
  Struct node *temp=*head_ref;
  if(position==0)
  {
     *head_ref=temp->next;
     free(temp);
     return;
  }
  for(int i=0; temp!=Null && i<position-1; i++)
      temp=temp->next;
  if(temp==Null || temp->next==Null)
      return;
  Struct node *next=temp->next->next;
  free(temp->next);
  temp->next=next;
}
  void printlist(struct node *node)
  {
    while(node!=Null)
    {
      printf("%d", node->data);
      node->next;
    }
  }
```

Scanned with CamScanner

```
int main()
{
 Struct Node *head =NULL;
  push(& head, 7);
  push(& head, 1);
  push(& head, 3);
  push(& head, 2);
  push(& head, 8);
 puts("created linked list:");
 printlist (head);
 delete Node(& head, 4);
 puts("\n linked list after deletion at
          position 4:");
 printlist(head);
 return 0;
}
```

Output:-

```
Created linked list:
  8 2 3 1 7
linked list after deletion at position 4:-
  8 2 3 1
```

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}

code:-

```
#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");
}
```

Scanned with CamScanner

```
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(structnode *a, structnode *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};
 int n=size of(keys)/size of(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;
}
```

out put:-

```
first list : 1 → 2 → 3 → Null
Second list : 4 → 5 → 6 → Null
After merge : 1 → 4 → 2 → 5 → 3 → 6 → null.
```

4. write a program to print elements in a Queue

(i) in reverse order?

Code:-

```
 # include<stdio.h>
 using name Space Std;
 void print (Queue<int>& Queue)
 {
  while(! Queue.empty()) {
     cout << Queue.front () << " ";
     Queue. pop();
  }
 }
```

```
void reverse Queue(Queue<int>& Queue)
{
  stack<int> stack;
  while(!Queue.empty()){
     Stack.push(Queue.front());
     Queue.pop();
  }
   while(!stack.empty()){
      Queue.push(stack.top());
      Queue.pop();
   }
 }
int main()
{
  queue<int> Queue;
  Queue.push(10);
  Queue.push(20);
  Queue.push(30);
  Queue.push(40);
  Queue.push(50);
  Queue.push(60);
  Queue.push(70);
reverseQueue(Queue);
print(Queue);
}
```

output:-

```
70, 60, 50, 40, 30, 20, 10
```

Scanned with CamScanner

ii. In alternate order

Code:-

```
#include<stdio.h>
#include<stdlib.h>
Struct node{
    int data;
    struct node* next;
};
void printAlternateNode(struct node* head)
{
    int count=0;
    while (head!=null){
        if(count%2==0)
            printf("%d", head->data);
        count++;
        head =head->next;
    }
}
void push(struct node** head_ref, int new_data)
{
    struct node* new_node=
      (struct node*)malloc(sizeof(struct node));
    new-node -> data = new-data;
    new-node -> next = (* head_ref);
    (* head_ref) = new_node;
}
int main()
{
    struct node* head = null;
    push(& head, 12);
    push(&head, 29);
    push(& head, 11);
    push(& head, 23);
    push(& head, 8);
    printAlternateNode(head);
    return 0;
}
```

output :-

```
8 11 12
```

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

Answer:-

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 where each node consists of the data and the references to the previous and next element.

Basically, an array is a set of similar data objects stored in sequential memory locations under a common heading or a variable name.

While a linked list is a data structure which contains a sequence of the elements where each element is linked to its next element. There are two fields in an element of linked list. One is data field, and other is link field, Data field contains the actual value to be stored and processed. Furthermore, the link field holds the adress of next data item in the linked list. The address used to access a particular node is known as pointer.

(ii) write a program to add the first element of one list to 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.

code:-

```
# include<stdc++.h>
using namespace std;
int maxsum (int ~~saf~~ stack1[], int stack2[],
        int stack3[], int n1, int n2, int n3)
{
int sum1=0, sum2=0, sum3=0;
for (int i=0; i<n1; i++)
    sum1 += stack1[i];
for (int i=0; i<n2; i++)
    sum2 += stack2[i];
for (int i=0; i<n3; i++)
    sum3 += stack3[i];
int top1=0, top2=0, top3=0;
int ans=0;
while(1)
{
    if(top1==n1 || top2==n2 || top3==n3)
    return 0;
```

```
if(sum1== sum2 && sum2== sum3)
return sum1;
if (sum1>= sum2 && sum1 >= sum3)
  sum1 -= stack1[top1++];
else if (sum2>= sum3 && sum2>= sum3)
  sum2 -= stack2[top2++];
else if(sum3>=sum2 && sum3>=sum1)
  sum3 -= stack3[top3++];
int main()
{
   int stack1[] = {3,2,1,1,1};
   int stack2[] = {4,3,2};
   int stack3[] = {1,1,4,1};
int n1 = sizeof(stack1)/sizeof(stack1[0]);
int n2 = sizeof(stack2)/sizeof(stack2[0]);
int n3 = sizeof(stack3)/sizeof(stack3[0]);
cout << maxSum(stack1, stack2, stack3, n1,
          n2, n3) << endl;
return 0;
}
```

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

code:-

```
#include<stdio.h>
int subArraysum(int arr[], int n, int sum)
{
 int cur_sum=arr[0], start=0, i;
 for (i=1; i<=n; i++)
 {
  while(curr_sum>sum&&start<i-1)
  {
   curr_sum=curr_sum-arr[start];
   start++;
  }
  if(curr_sum==sum)
  {
  printf("sum found between indexes %d and
         %d", start, i-1);
  return 1;
  }
  if (i<n)
  curr_sum=curr_sum+arr[i];
 }
 printf("No subarray found");
 return0;
}
```

```
int main()
{
int arr[] = {15, 2, 4, 8, 9, 5, 10, 23}
int n = size of (arr) / size of (arr[0]);
int sum = 23
SubArray sum (arr, n, sum);
return 0;
}
```

Scanned with CamScanner