Assignment :-

N. Vaishnavi
AP19110010487
CSE-G.

① program to insert and delete an element at n^{th} & k^{th} position in linked list.

```
#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 print_linked_list();
void insert_at_list(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");
    created_linked_list();
    print_linked_list();
    printf("\n insert new item at last \n");
    scanf("%d", & value);
```

Scanned with CamScanner

```
    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 from list \n");
    scanf("%d", &value);
    search_item (value);
//Delete value from linked list.
    printf("\n Enter a value, which your want to delete \n");
    scanf("%d", &value);
    delete_item (value);
    print_linked_list();
     return 0;
}
/*
   user defined functions
*/
```

Scanned with CamScanner

```
void create_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 [size of (node)],
  temp_node -> number = value;
  temp_node -> next = Null;
  // for the 1st element
  if(head == Null)
  {
     head= temp_node;
     last = temp_node;
  }
  else
  {
     lost -> next = temp_node;
     lost = temp_node;
  }
}
Void insert_at_first(int value)
{
```

Scanned with CamScanner

```
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 (mynode -> Number == key)
{
node *new node = (node*) malloc (size of (node));
new node -> Number = value;
new node -> Next = my node -> next;
my node -> Next = new node;
printf ("%d is inserted after %d \n", value, key);
flag = 1;
break;
}
else
my node = my nod -> Next;
}
if ( flag == 0)
printf ("key not found ! \n");
```

Scanned with CamScanner

```
void delete_item (int value)
{
 node * mynode = head; * previous = Null;
 int flag = 0;
  while (mynode != Null)
  {
   if (mynode → Number == value)
    {
      if (perivous == Null)
          head = mynode → next;
      else
        perivous → 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;
```

Scanned with CamScanner

```
while (my list != null)
  {
    printf ("%d", mylist -> Number);
     mylist = mylist -> Next;
  }
  puts (" ");
}
```

Output :——

Create linked list
Input a number (enter -1 to exist)
1
Input a number (enter -1 to exist)
2
Input a number (enter -1 to exist)
3
Input a number (enter -1 to exist)
4
Input a number (enter -1 to exist)
5
Input a number enter -1 to exisf)
-1
your full linked list is
12345
Insert new item at ~~f~~ist.
123456.
Insert new item at first
0

Scanned with CamScanner

your full linked list is

0123456

enter a key (existing item of list),

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

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.

Scanned with CamScanner

⑤ Different b/w Array & Linked list ?

Array	Linked list
1) An array is a collection of elements of a similar data type	1) Linked list is an ordered collection 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 in elements will have to be accessed sequentially
3) Data elements are stored in contigous locations in memory	3) New elements can be stored any where and reference is created for the new element using pointers.

Scanned with CamScanner

5) ii, Program

```
#include <stdio.h>
#include <stdlib.h>
int len(int a[])
{
  int i=0, a, n=0;
  while (1)
  {
    if(a[i])
    {
      an++; i++;
    }
    else
    {
      break;
    }
  }
  return an;
}
void changing list (int a[], int c[])
{
  for(int i=len(a)-1; i>=0, i--)
  {
    a[i+1]=a[i];
  }
    a[0]=c[0];
    printf("\n the elements of first array:\n");
  for (int i=0; i<len(a); i++)
  {
    printf("%d", a[i]);
  }
  for (int i=0, i< len(a); i++)
```

Scanned with CamScanner

```
{
  c[i] = c[i+1]; }
  printf("In the elements of second array :\n");
  for (int i=0; i<len(c); i++)
  {
    printf("%d ", c[i]);
  }
}
int main()
  {
    int a[10] = {1,2,3} , c[10] = {4,5,6},
      Changing list (a,c);
  }
```

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

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int data;
  struct node *next;
};
void print rev(struct node *head)
{
  if (head == Null)
    return;
    print rev(head -> next);
    printf("%d ", head -> data);
void push (struct node *headrev, char new)
{
  struct node *node_new = (struct node *) malloc (size of (struct
                                                    node));
```

Scanned with CamScanner

```
    node_new -> data = new;
    node_new -> next = (head *_ref);
    (*head ref) = node_new;
}
int main()
 struct node * head = Null;
  push(& head, 4);
  push(& head, 3);
  push (& head, 2);
 print new(head); print alternate (head);
 return 0;
}
void print alternate (struct node * head)
 {
   int count = 0;
   while (head != Null)
 { if (count % 2 == 0)
      count << head -> data << " ";
       count ++;
     head = head -> next;
 }
```

Scanned with CamScanner

③

```
#include <stdio.h>
int sum = 0;
int l=0,h=0;
for (l=0; l<n; l++){
  while (sum<s && h<n)
    sum += arr[h];
      h++;
if (summ==s)
{
  printf ("found");
   return ; }
   sum == arr[l];
}
}
int main (void) {
  int arr=[] = {2,6,0,9,9,3}
  int a=15;
   int n = sizeof (arr)/sizeof (arr[0]);
   find (arr, n,s);
   return 0;
}
```

Scanned with CamScanner

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

```
#include <stdio.h>
#include <stdlib.h>
// Data structure to 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 * new node = (struct node *) malloc (sizeof(struct node));
   new node → data = data;
   new node → next = * head;
      *head = new node;
}
// Function to construct a linked list by merging alternate
node
```

Scanned with CamScanner

```
// two given linked lists using dilbar node.
struct node * shuffle merge(struct node * c, struct node * d)
{
  struct node dilbar;
  struct node* tail = & dilbar;
  dilbar.next = Null;
  while (1)
  {
    // empty list cases.
    if(c == = Null)
    {
      tail→Next = d;
      break;
    }
    else if (d == = Null);
    {
      tail→ Next = c;
      break;
    }
    // move two nodes to tail
    else
    {
      tail → next = c;
      tail = c;
      c = c→ Next;
      tail→ next = d;
      tail = d;
      d = d→ Next;
    }
  }
  return dilbar.next;
}
int main (void)
{
```

Scanned with CamScanner

```
int keys[] = { 1, 2, 3, 4, 5, 6, 7 }
int n = sizeof(keys)/sizeof(keys[0]);
struct node * c = Null, * d = Null;
for (int j = n-1; j >= 0; j = j-2)
    push(&c, keys[j]);
for (int j = n-2; j >= 0; j = j-2)
    push(&d, keys[j]);
printf("Frist list: ");
Print list(c);
printf("second list:");
Printlist(d);
Struct node *head = shuffle merge(c,d);
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.

Scanned with CamScanner

(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. Exist");
  do
   {
     printf("\n enter the choice : ");
     scanf("%d", &choice);
     switch (choice)
     {
       case 1:
         {
           push()
           break;
         }
       case 2:
         {
           display()
           break;
         }
       case 3:
         {
           Subarray sum()
           break;
         }
       case 4;
         {
           printf("\n\t Exist point");
           break;
         }
```

Scanned with CamScanner

```
        default;
        {
          printf("\nIt plz enter a valid choice (1/2/3/4)");
        }
    }
  }
  while (choice != 4)
  return 0;
}
void push()
 {
  if (top >= n-1)
   {
     printf("\nIt stack if over flow");
   }
  else
   {
     printf("enter the value");
     scanf("%d", &v);
     top++
     stack[top] = v;
   }
  }
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");
   }
  else
   {
     printf("\n the stack is empty");
   }
```

Scanned with CamScanner

```
}
int subarraysum(int stack[], intsum)
{
  int curr_sum, i, k;
  scanf("%d", &sum);
  for (i=0; i<n; i++)
  {
    curr_sum = stack[i]; stack[i]);
  // try all subatrays starting with i
  for (k= i+1, k<=n, k++)
    {
      if (curr_sum == sum)
       {
        printf("sum found %d and %d, i, stack[i], stack[i]);
       return 1;
      }
      if (curr_sum > sum || k==n)
          break;
     curr_sum = curr_sum + stuck[k];
    }
  }
   printf("No subarray found");
   return 0;
  }
  int main()
  {
     int sum = 23;
     Sub array sum (stack ,n,sum);
     return 0;
  }
```

Scanned with CamScanner

Output :-

1. push
2. display
3. Subarray
4. Exit

Enter choice : 1
Enter a value
1
Enter choice = 1
Enter a value.
2
Enter choice = 1
Enter a value.
3
Enter choice = 1
Enter a value
4
Enter choice : 2
the element in stack
1
2
3
4
Press next choice :
3
sum found 1, 2

Scanned with CamScanner