①

T. Venkata Nagalakshmi
AP19110010223
CSE - G.

Assignment- 1

1. Write a program to insert and delete an element at the nth and kth position in a linked list where n and k is taken from user.

```
# include <stdio.h>
# include <stdlib.h>
struct node
{
    int data;
    struct node* next;
};

int main()
{
    int i, n, item, pos, dpos;
     struct node *p, *q, * head, * newnode,* nextnode;
    printf(" enter the no. of nodes you want: ");
    scanf("%d", &n);
    printf(" enter the value for the head node: ");
    scanf(" %d", & item);
    int count = 1;
    q = (struct node *)
    malloc (size of (struct node));
    q -> data = item;
    q -> next = NULL;
    head = q;
    p = head;
    for(int i=1; i<n; i++)
    {
      printf(" enter the value for the next node: ");
      scanf(" %d", & item);
      q= (struct node*)
```

```
malloc (size of (struct.node));
 q -> data = item;
 q -> next = NULL;
  count ++;
 p -> next = q ;
 p = p-> next;
}
p = head;
while (p! = NULL)
{
   printf(" the elements before inserting an element: %d\n",
                      p->data);
    p = p -> next;
 }
printf(" enter the position you want to enter element: ");
scanf(" %d", & pos );
int j=1
p = head;
if (pos >n)
{
    printf(" Invalid choice \n ");
     p = head;
  }
else {
    while (j<pos)
 {
    p=p->next;
     j++;
 }
    newnode = (struct node*) malloc (size of (struct node));
  printf(" enter the value of new node: ");
```

```
        Scanf(" %d ", &newnode->data);
        newnode->next= p->next;
        p-> next = new node;
        count ++;
    }
    p=head;
    while (p! =NULL)
    {
        printf(" the elements after inserting an element: %d\n",
                    p->data);
          p=p->next;
     }

   Printf(" enter the position you want to delete element: ");
   Scanf (" %d ", &dpos);
   p=head;
   if ( dpos > count)
   {
       printf (" Invalid \n");
   }
  else {
      int k=1;
      p= head;
      while (k<=pos)
    {
        nextnode = p;
        p= p->next;
        k++;
    }
    next node->next = p->next;
    printf (" deleted element: %d\n", p->data);
    free (p);
```

```
        }
        p= head;
        while (p! = NULL)
        {
          printf (" the elements after deleting an element: %d
                    \n", p-> data);
          p = p-> next;
        }
}
```

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 >
struct node
{
   int data ;
   struct node* next ;
};
int main ()
{
   int i, n, item1, item2, m;
   struct node *p, *q, * head1 * , * head 2 , * r, * s, * temp, *a1, *b1;
   printf (" enter no of nodes you want for first list: ");
   scanf (" %d ", & n);
   printf(" enter the value for the head node: ");
   scanf (" %d ", & item1);
    q = (struct node *)
   malloc (size of (struct node));
```

```
q -> data = item 1 ;
q -> next = NULL ;
head 1 = q ;
p = head 1 ;
for (int i=1; i<n; i++)
{
  printf(" entee the value for the next node : ");
  scanf (" %d ", & item 1 );
  q = (struct node *)
 malloc (size of (struct node));
   q -> data = item 1 ;
   q -> next = NULL ;
    p -> next = q ;
   p = p -> next ;
}
p = head 1 ;
printf (" entee no of nodes you want for second list : ");
scanf (" %d ", & m);
printf(" entee the value for the head node : ");
scanf(" %d ", & item 2 );
s = (struct node *)
malloc (size of (struct node));
s -> data = item 2 ;
s -> next = NULL ;
head 2 = s ;
r = head 2 ;
for (int i=1; i<m; i++)
{
  printf(" entee the value for the next node : ");
  scanf(" %d ", & item 2 );
```

```
        s = (struct node *)
    malloc (size of (struct node));
        s -> data = item 2 ;
        s -> next = NULL ;
        r -> next = s ;
        r = r -> next ;
    }
    r = head 2 ;
    temp = p;
    while (p! = NULL && q! = NULL)
    {
        a1 = p -> next ;
        b1 = r -> next ;
        p -> next = r;
        r -> next = a1 ;
        p = a1 ;
        r = b1 ;
    } .
    while (temp! = NULL)
    {
        printf(" the alternate merged linked list values: %d
                 \n", temp -> data);
        temp = temp -> next ;
    }
}
```

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

```
# include < stdio. h>
int main()
```

```
{
  int n, stack [100], top= -1, x, k, sum, y;
  printf(" enter no of elements you want to enter in to stack:");
  scanf ("%d", &n);
  for (int i=0; i<n; i++)
  {
    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;
    }
  }

  printf(" enter the sum you want for:");
  scanf (" %d ", &k);
  if (top< =-1)
  {
     printf(" \n It stack is underflow ");
  }
  else
  {
    while (top! = -1)
     {
       y= stack [top];
       sum=0;
       while (y! =0)
       {
```

```
                sum = sum+ (y % 10);
                y= y/10;
              }
              if (sum = = k)
              {
                printf(" the element  is %d\n", stack [top]);
              }
              else
              {
                printf(" element  not  found \n");
              }
              top --;
            }
        }
    }
```

4. Write a program to print the elements in a queue.
1) in reverse order
2) in alternate order.

```
# include < stdio. h >
# define  MAX 100
int queue _ arr[max];
int stack [100] ;
int top = -1 ;
int rear = -1;
int front = -1;
int main()
{
    int n ;
    printf(" enter no of elements  you  want  to  enter  into  the
        queue:");
        scanf(" %d ", & n);
        for (int i=0; i<n; i++)
        {
          int add_item;
          if (rear = = max-1)
          printf(" Queue Overflow \n ");
        else
         {
           if (front = = -1)
           front = 0;
           printf(" inset the element in queue : ");
           scanf(" %d ", & add_item);
           rear = rear+1;
           queue_arr[rear] = add_item;
       }
    }
    int choice;
    printf(" 1. to print elements in a reverse order \n
              2. to print elements in a alternate order \n");
    printf(" enter your choice: ");
    scanf(" %d ", & choice);
    switch (choice)
     {
       case 1:
        {
          for (int i=0; i<n; i++)
           {
           if (front = = -1 || front > rear)
             {
                printf(" Queue underflow \n");
             }
           else
             {
                top ++;
                stack [top] = queue_ arr[front];
                front = front + 1 ;
              }
           }
           if (top>=0)
            {
                printf(" \n the elements in queue in reverse
                        order \n");
                for (int i=top ; i>=0; i--)
               printf(" %d\n ", stack [i]);
                printf ("\n ");
             }
            else
             {
                printf(" \n the stack is empty");
             }
            break ;
       }
     case 2 :
     {
        for (int i = front; i<=rear, i+=2)
        printf("%d\n", queue_ arr[i]);
        break ;
     }
     default ;
      printf(" enter a valid choice: ");
      break ;
    }
}
```

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

Key differences b/w array and linked list.

1. An array is a data structure that contains a collection of similar type data elements whereas the linked list is considered as non-primitive datastructure contains a collection of unordered linked elements known as nodes.

2. In the array the elements belong to indexes, i.e.., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.

3. In a linked list through, you have to start from the head and work your way through until you get to the fourth element.

4. According all element in an array is fast, while in linked list takes linear time, so it is quite a bit slower.

5. operations like insertion and deletion in array consume a lot of time. On the other hand the performance of these operations. in linked list is fast.

6. In a array, memory is assigned during compile time while in linked list it is allocated. during execution of runtime.

(ii) Write a program to add 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.

```
#include <stdio.h>
# include <stdlib.h>
int len (int a[])
{
    int i=0, an=0;
     while (1)
     {
       if (a[i])
```

```
            {
                an++ ; i++;
            }
            else
              {
                  break;
              }
            }
            retuen an;
    }
    void  changinglist (int a[], int b[])
      {
          for (int i= len(a) -1 ; i>=0; i--)
          {
             a[i+1] = a[i];
          }
        a[0] = b[0];
       printf(" the elements of first aray : \n ");
      for (int i=0; i<len(a) ; i++)
        {
          printf("%d", a[i]);
        }
       for (int i=0; i<len(b) ; i++)
         {
            b[i] = b(i+1) ;
         }
         printf(" the elements of second array: \n ");
         for (int i=0 ; i<len(b) ; i++)
          {
            printf (" %d", b[i]);
          }
     }
     int main()
       {  int a [10] = {1,2,3} , b [10] = {4,5,6} ;
          changinglist (a,b);
       }
```