DSA Assignment - 4

Jaya Harshith. A
AP19110010515
CSE - H

1. Write a c program to insert and delete an element at the n^{th} and K^{th} position in a linked list where n and K is taken from the user.

Solution:-

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
struct node
{
    int my_data;
    struct node * next;
};
struct node * head = NULL;
void insert (int n)
{
    struct node * newnode;
    int i;
    newnode = (structnode *) malloc (size of (structnode *));
    printf ("Enter my_data = ");
    scanf ("%d", & newnode -> my_data);
    if (n == 1)
    {
        newnode -> next = head
        head = newnode;
    }
    else
    {
        struct node *temp = head;
        for (i = 1; i <= n; i++)
            temp = temp -> next;
```

Scanned with CamScanner

```
        newnode -> next = temp -> next;
        temp -> next = newnode;
    }
}
void delete (int k)
{
    struct node * temp1 = head;
    if (k==1)
    {
        head = temp1 -> next;
        free (temp1);
    }
    else
    {
        int i.
        for(i=1; i<k-1; i++)
            temp1 = temp1 -> next;
        struct node * temp2 = temp1 -> next;
        temp1 -> next = temp2 -> next;
        free (temp2);
    }
}
void display ()
{
    struct node * newnode;
    newnode = head;
    printf ("Linked list =\n");
    while (newnode != NULL)
    {
        printf ("\n%d", newnode -> data);
        newnode = newnode -> next;
    }
}
```

Scanned with CamScanner

```
void main()
{
    int n, k, ch;
    while(1)
    {
        printf("1. Insert \n 2. Delete \n 3. Display \n 4. Exit");
        printf("Enter your choice =");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: printf("Enter position to insert =");
                    scanf("%d", &n);
                    insert(n);
                    break;
            case 2: printf("Enter position to delete =");
                    scanf("%d", &k);
                    delete(k);
                    break;
            case 3: display()
                    break;
            case 4: exit(0);
            default: printf("Input is wrong");
        }
    }
```

Scanned with CamScanner

output of the program:

```
1. Insert
2. Delete
3. display
4. Exit
Enter your choice = 1
Enter position to insert = 1
Enter my_data = 5
Enter your choice = 1
Enter position to insert = 2
Enter my_data = 10
Enter your choice = 3
5
10
```

Scanned with CamScanner

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

Solution:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
struct node
{
    int my_data;
    struct node * next;
};
void insert_at_begin (struct node **head, int my_data)
{
    struct node * newnode = (struct node *) malloc (size of (struct node));
    newnode->mydata
    newnode -> my_data = my_data;
    newnode -> next = *head;
    *head = newnode;
}
void display (struct node * head)
{
    struct node * temp = head;
    while (temp)
    {
        printf ("%d ->", temp -> my_data);
        temp = temp -> next;
    }
}
```

Scanned with CamScanner

```
void merge (struct node **x, struct node **y)
{
    struct node temp;
    struct node *tail = &temp;
    temp.next = NULL;
    while (1)
    {
        if (*x == NULL)
        {
            tail -> next = NULL;
            break;
        }
        else if (*y == NULL)
        {
            tail -> next = *x;
            break;
        }
        else
        {
            tail -> next = *x;
            tail = *x;
            *x = (*x) -> next;
            tail -> next = *y;
            tail = *y;
            *y = (*y) -> next;
        }
    }
    *x = temp.next;
}
```

Scanned with CamScanner

```
void main()
{
    int i, a, b, m, n;
    struct node * list1 = NULL, * list2 = NULL;
    printf ("Enter number of elements to insert to list1 =");
    scanf ("%d", &a);
    for (i=1; i<=a; i++)
    {
        printf ("Enter the element to insert = ");
        scanf ("%d", &m);
        insert_at_begin (&list1, m);
    }
    printf ("First list = \n");
    display (list1);
    printf ("Enter number of elements to insert for list2 =");
    scanf ("%d", &b);
    for (i=1; i<=b; i++)
    {
        printf ("Enter element to insert = ");
        scanf ("%d", &n);
        insert_at_begin (&list2, n);
    }
    printf ("second list = \n");
    display (list2);
    merge (&list1, &list2);
    printf ("After merging = \n");
    printf ("Merged list is = \n");
    display (list1);
}
```

Scanned with CamScanner

output of the program:

Merged list is = 1 4 2 5 3 6

Scanned with CamScanner

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

Solution:

```
#include <stdio.h>
int top=-1;
int x;
char stack[100];
void push(int x);
char pop();
int main()
{
    int i,n,a,t,K,f,sum=0, count=1;
    printf("Enter number of elements in the stack=");
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        printf("Enter next element=");
        scanf("%d",&a);
        push(a);
    }
    printf("Enter the sum to be checked=");
    scanf("%d",&K);
    for(i=0; i<n; i++)
    {
        t=pop();
        sum+=t;
        count+=1;
        if(sum==K)
        {
            for(int j=0; j<count; j++)
                printf("%d", stack[j]);
            f=1;
```

Scanned with CamScanner

```
            break;
        }
        push(t);
    }
    if (t!=1)
        printf("The elements in the stack dont add upto sum");
}
void push(int x)
{
    if (top==99)
    {
        printf("\n Stack is full\n");
        return;
    }
    top = top+1;
    stack[top]=x;
}
char pop()
{
    if (stack[top] == -1)
    {
        printf("\n Stack is empty\n");
        return 0;
    }
    x = stack[top];
    top = top-1;
    return x;
}
```

Scanned with CamScanner

output of the program :

Enter the number of elements in the stack = 5

Enter next element = 1

Enter next element = 2

Enter next element = 3

Enter next element = 4

Enter next element = 5

Enter the sum to be checked = 5

2 3

Scanned with CamScanner

9. Write a program to print the elements in a queue.

(i). in reverse order.

(ii). in alternate order.

Solution:

```
#include <stdio.h>
#include <stdlib.h>
#define size 20
int queue[20], front = -1, rear = -1;
void enqueue (int k)
{
    if (rear == size-1)
        printf ("Queue is full");
    else
    {   printf ("Enter the k = ");
        scanf ("%d", &k);
        if (front == -1)
            front = 0;
        rear++;
        queue [rear] = k;
    }
}
void dequeue ()
{
    if (front == rear)
        printf ("Queue is empty");
    else
    {
        printf ("\n deleted : %d", queue [front]);
        front++;
        if (front == rear)
            front = rear = -1;
    }
}
```

Scanned with CamScanner

```
void displayreverse()
{
    if (rear==-1)
        printf("Queue is empty");
    else
    {
        int i;
        printf("\n Queue is :\n");
        for (i=rear; i>=front; i--)
            printf("%d", queue[i]);
    }
}
void displayalternate()
{
    if (rear==-1)
        printf("Queue is empty");
    else
    {
        int i;
        printf("Queue alternate elements are \n");
        for (i=front; i<=rear; i++)
            printf("%d", queue[i]);
    }
}
```

Scanned with CamScanner

```
void main ()
{
    int k, choice;
    while (1)
    {
        printf ("1.Enqueue\n 2. Dequeue\n 3. Display reverse\n
                 4. Display alternate list\n");
        printf ("Enter your choice =");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1: printf ("Enter element to insert =");
                    scanf ("%d", &k);
                    enqueue (k);
                    break;
            case 2: dequeue ();
                    break;
            case 3: display reverse ();
                    break;
            case 4: display alternate ();
                    break;
            case 5: exit (0);
            default: printf ("Input is wrong or
                              choice is wrong");
        }
    }
}
```

Scanned with CamScanner

output of the program:

1. Enqueue
2. Dequeue
3. Display reverse
4. Display alternate
5. Exit

Enter your choice = 1

Enter K = 2

1. Enqueue
2. Dequeue
3. Display reverse
4. Display alternate
5. Exit

Enter your choice = 1

Enter K = 3

1. Enqueue
2. Dequeue
3. Display reverse
4. Display alternate
5. Exit

Enter your choice = 1

Enter K = 5

1. Enqueue
2. Dequeue
3. Display reverse
4. Display alternate
5. Exit

Enter your choice = 4

~~Queue elements are =~~ Queue alternate elements are =

25 1. Enqueue
2. Dequeue
3. Display reverse
4. Display Alternate
5. Exit

Scanned with CamScanner

Enter your choice = 3

Queue is =

5

3

2 1. Enqueue

2. Dequeue

3. Display reverse

4. Display alternate

5. Exit

Enter your choice = 5

Scanned with CamScanner

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

(ii). Write a program to add the first element of one list to another list. ~~for exam~~

(i). Solution:

Array:	Linked list:
• An array is a collection of elements stored in adjacent memory locations.	• Linked list is an order collection of nodes which have two parts data, next. Nodes are elements connected by pointers.
• Array uses static memory allocation. (that is fixed memory).	• Linked list uses dynamic memory allocation. (memory allocated at execution)
• Size of the array must be specified during initialisation.	• Size of the linked list is adjusted according to insertion, deletion.
• Random access is possible in array	• Random access is not possible in linked list. Elements can be accessed orderly.
• No memory waste if the array is full or almost full. Otherwise may result in much memory wastage.	• Since memory is allocated dynamically, there will be no waste of memory.

Scanned with CamScanner

(ii) Solution :

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
struct node
{
    int my_data;
    struct node *next;
};
void insert_at_begin (struct node **head, int my_data)
{
    struct node *new_node = (struct node *) malloc
                            (sizeof(struct node));
    new_node -> my_data = my_data;
    new_node -> next = *head;
    *head = new_node;
}
void move_node (struct node **destination, struct node **source)
{
    if (*source == NULL)
        return;
    struct node *new_node = *source;
    *source = (*source) -> next;
    new_node -> next = *destination;
    *destination = new_node;
}
```

Scanned with CamScanner

```
void display (struct node * head)
{
    struct node * temp = head;
    while (temp)
    {
        printf ("%d →", temp → my_data);
        temp = temp → next;
    }
}
void main()
{
    int n1, n2, a, b, i;
    printf ("Enter number of nodes in list 1 =");
    scanf ("%d", &n1);
    struct node * list1 = NULL;
    for (i=1; i<=n1; i++)
    {
        printf ("Enter element to insert into list 1 =");
        scanf ("%d", &a);
        insert_at_begin (&list1, a);
    }
    printf ("Enter number of nodes in list 2 =");
    scanf ("%d", &n2);
    for (i=1; i<=n2; i++)
    {
        printf ("Enter element to insert into list 2 =");
        scanf ("%d", &b);
        insert_at_begin (&list2, b);
    }
    movenode (&list1, &list2);
    // Printing after moving node
```

Scanned with CamScanner

```
printf ("First list = \n");
display (list1);
printf ("\n Second list is = \n");
display (list2);
}
```

output of the program:

Enter number of nodes in list1, list2 = 2, 3

First list = 3 -> 2 -> 1 -> Second list = 2 -> 1 ->

Scanned with CamScanner