DSA-ASSIGNMENT-2

VENKATA MANASS API91600 18586

1)

```
/* Program to insert and delete an element at the nth and kth position in a linked list where n and k are taken from user. */

#include <stdio.h>
#include <stdlib.h>

struct node {
int data;
struct node * next;
};

struct node* head;
void insert (int n, int data)
{
Node* temp = new node ();
temp -> next = Null;
if (n==1)
{
temp->next=head;
head=temp;
return;
}
void delete
{
Stut Node* temp = head;
if (k==1)
{
head = temp -> next;
return;
}
Node* temp = head;
```

```
for (int j=0 ; j<n-2 ; j++)
 temp = temp->next;
 }
 temp->next = temp->next;
 temp->next = temp;
 }
 void print()
 for (int j=0 ; j<k-2 ; j++)
 temp = temp->next;
 free(temp);
 }
 int main()
 {
 int n,a,k;
 head = NULL;
 Printf("[redacted] the position for inserting
 scanf("%d",&n);
 scanf("%d",&a);
 insert(a,n);
 Printf("Give [redacted] the position for deleting
 scanf("%d",&k);
 delete(k);
 Printf("\n");
 return;
 }
```

2)

/* C Program to construct a new linked list by merging alternative nodes and two lists */

Sol:-

```
#include <stdio.h>
#include <stdlib.h>
struct node {
 int data;
 struct node *next;
}
Void Print list (struct node *head)
{
 Printf ("%d ->", ptr->data));
  ptr = ptr->next;
  Print f ("Null \n");
  }
Void Push (struct node *head, int data)
{
Struct node* new = (struct node)
malloc (sizeof of (struct node));
new->data = data;
new->next = *head;
 *head = new;
}
```

```
void main()
{
int keys[] = {1,2,3,4,5,6,7};
int n = sizeof(keys)/
struct node* l = Null; *m = Null;
for(int j = n-1; j>0; j = j-2)
   Push(&l, keys[j]);
for(int j = n-2; j>=0; j = j-2)
   Push(&m, key[j]);
struct node*head = merge(l,m);
Print list(head);
}
```

30

```
/*C program to find all the elements in
  the stack whose sum is equal to K*/

#include<stdio.h>
 int top=-1;
 int a;
 char stack[100];
 void push(int a);
 char pop();
 int main()
 {
 int i,n,x,y,j,k,sum=0,count=1;
 printf("Give the no. of elements");
 }
```

```
struct node *merge (struct node *a, struct node *b)
{
struct node false;
struct node *fail = &false;
false.next = NULL;
while (1)
{
if (a == NULL)
{
fail->next = b;
break;
}
else if (b == NULL)
{
fail->next = a;
break;
}
else
{
fail->next = a;
fail = a;
a = a->next
fail->next = b;
b = b->next;
}
}
return false.next;
}
```

```
scanf("%d", &n);
for(j=0; j<n; j++){
printf("Give the next element ");
scanf("%d", &x);
push(x);
}
printf("Provide the sum toverify");
scanf("%d", &k);
for(j=0; j<n; j++)
{
y= pop();
sume = sum+y;
cout = count+1;
if(sum==k){
for(int i=0; i<count; i++)
printf("%d", stack[i]);
a=1;
break;
}
push(y);
}
if(a!=1)
printf("It is not equal to the sum");
}
void push(int a)
{
if(top==2){
printf("\n stack is full");
return;
```

```
top= 1 ;
stack [top]= a;
}
char pop()
{
if (stack[top]==-1)
{
printf ("The stack is fully empty");
return 0;
}
a = stack[top];
top= top-1
return a;
}
```

---

10. /*C program to print the elements in
queue i. in reverse order
ii. in alternate order.

```
#include <stdio.h>
#define size 20
void insert(int);
Void delete();
int queue[20], x=-1, y=-1;
Void main()
{
int value, choice;
while(1)
{
Printf("1.Insertion");
Printf("2. Deletion");
Printf("3. Print reverse");
Printf("4. Print Alternative");
Printf("5. Exit");
Printf("Chose one option");
Scanf("%d", &choice);
&
```

```
switch (choice)
{
case 1:
printf("Value to insert: ");
scanf("%d", &value);
insert(value);
break;
case 2:
[illegible]
delete();
break;
case 3:
printf("The reversed que is");
for(int j=size; j>=0; j--)
{
if(queue[j]==0)
continue;
Printf("%d", queue[j]);
}
break;
case 4:
printf("Alternate elemet is");
for(int j=0; j<size; j+=2)
{
if(queue[j]==0)
continue;
printf("%d", queue[i]);
}
break;
case 5:
exit(0);
default:
printf("Wrong option");
}
```

```
void insert(int value){
if ((x==0&&y==size-1)|| x==y+1)
Printf("\n the insertion is not at all posible");
else{
if (x==-1)
x =0;
y=(y+1)%size;
queue[y]=value;
Printf("\n Insrtion is done");
}
}
Void delete(){
if (x==-1)
Printf("deletion is not at all posible");
else{
Printf("%d", queue[x]);
x=(x+1)%size;
if(x==y)
x =y=-1;
}
}
```

5)

i. How array is different from the linked list.

The Major differencese betwee n array and linked list are:-

i) Arrays are index based data structure where each element is assosiated with an index. And the linked list relies on references where each node consists of the data and the reference to the

previous and next element.

99) Elements are sorted consecutively in array whereas it is sorted randomly in linked list.

99) /* c program to add the first element of one list to another list */

```
#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_data = (struct node*
 malloc(sizeof(struct node));
new_node->data = new_data;
new_node->next = (*head_ref);
(*head_ref) = new_node;
}
void printList(struct node* head)
{
while(temp != NULL)
{
printf("%d", temp->data);
temp = temp->next;
}
```

```
void merge(struct node** x, struct node** y);
{
struct node* x_curr = *x, *y_curr = *y;
struct node* x_next = *x, *y_next;
while (x_curr != NULL && y_curr != NULL)
{
    x_next = x_curr->next;
    y_next = y_curr->next;
    y_curr->next = x_next;
    x_curr->next = y_next;
    x_curr = x_next;
    y_curr = y_next;
  }
  *y = y_curr;
}
int main()
{
struct node *x = NULL, *y = NULL;
push(&x, 1);
push(&x, 2);
push(&x, 3);
printf("The only first linked list is:");
printList(x);
push(&y, 4);
push(&y, 5);
push(&y, 6);
printf("second linked list");
printList(y);
merge(&x, &y);
printf("Modified first linked list:");
printList(x);
printf("Modified second linked list:");
printList(y);
}
```

```
merge(x, &y);
Printf("corrected first linked list\n");
Print list(x);
Printf("corrected second linked list\n");
Print list(y);
 getchar();
 return(0);
 }
```