Assignment-4

Ch. Avinash Reddy
AP19110010513.

(1)

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
int data;
struct node* next;
};
Struct Node* head;
void Insert(int data, int n){
Node* temp= new node();
temp->data =data;
temp->next = NULL;
if (n==1){
temp->next =head;
head = temp;
return;
}
void Delete_(int k){
struct Node* temp=head;
if(k == 1){
head=temp= next;
free(temp);
return;
}
Node* temp=head;
for(int i = 0; i<n-2; i++){
temp = temp->next;
}
temp = next= temp->next;
temp = next= temp;
}
```

```
voidprint();
for(inti=0,i<k-2,it+)
  temp=temp->next;
  free(temp);
 }
intmain(){
int n,x,k;
head=Nubl;
printf("enter position to and inserting:");
scanf("%d%d",&n&x);
Insert(x,n);
Printf("Enter position to delete);
scanf("%d",&k);
Delete(x);
Print(x)
 return;
}
```

2.
```
#include<stdio.h>
#include<stdlib.h>
Struct node{
    int data;
    Struct node*next;
  }
  Voidprintlist(Struct node*head)
  }
Printf("%d->",(ptr->data));
 ptr=ptr->next;
Printf("Null/n");
}
 Void push(struct node * head, int declare)
{
struct node *n=(struct node)malloc
          (sizeof structnode));
 n->data=data;
 n=> next=*head;
  *head>new.
}
struct node *merge(struct node*s,structnode* b)
}.
Structnode fake;
 struct node *tail=fake;
 fake.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;
}
}
return fail -> next;
}
void main()
{
int keys[] = {1,2,3,4,5,6,7}
int n = sizeof(keys)/sizeof(key[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, key[i]);
struct node * head = merge(a, b);
Print list (head);
}
```

3.
```
#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 no of elements in stack");
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Enter next element");
scanf("%d",&a);
push(a);
}
printf("Enter the sum to check if");
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[i]);
f=1)
break;
}
```

```
Push(i);
}
if (f!=1)
Printf("The elements are not resulting sum");
}
void push(int x)
{
if (top==99)
{
Printf("\n Stack is full\n");
return;
}
top=top+1;
stack(top)=x;
}
if(stack[top]==-1)
{
printf("\n stack is empty\n");
return 0;
}
x = stack(top);
top=top-1;
return x;
}
```

4 i)

```
# include <stdio.h>
# include "stack.h"
# include "QQ.h"
int main()
{
int n, a[20], i, i=0;
struct stack s;
int stack(s);
    printf("enter no");
    scanf("%d", &n);
 for(i=0; i<n, i++)
 {
printf("enter values:");
scanf("%d", &a[i]);
}
for(i=0; i<n; i++)
{
 insert(a[i]);
}
 while(i!=n)
{
 push(&s, def[]);
  i++;
}
printf("reverse is");
while(stop != -1)
{
 printf("%d", pop(&s));
}
printf("\n");
return 0;
}
```

ii)
```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node* next;
};
void print node(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);
}
int main()
{
struct node* head-null;
Push (&head, 12);
push(&head, 24);
push(&head, 11);
push(&head, 23);
Push(&head, 81);
~~push(&head, thread)~~
print node(head);
return 0;
}
```

5.

i) The major difference between array and linked list regards to their structure, arrays are index based data structures where each element is associated with an index on the other hand, linked list helleson of pointers to the previous and next elements.

ii)
```
#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_node = (struct node*) malloc (sizeof
                                    (struct node));

new_node -> data = new_data;
new_node -> next = (*head_ref);
(*head_ref) = new_node;
}
void print list (struct node * head)
{
struct node * temp = head;
while (temp != Null)
{
```

```
printf("%d ", temp->data);
temp= temp->next;
}
  printf("\n");
}
```