Assignment - 4

V. Vinod Varma
AP 19110010355

1)

```
# include < stdio.h >
# include < stdio.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 ) ( ) {
Struct  Node * temp = head ;
if ( x == 1 ) {
 head = temp → next ;
```

```
return;
}
Node * temp = head;
for (int i=0; i<n-2; i++){
temp = temp→ next;
}
temp→ next = temp→ next;
temp→ next = temp;
}
void print ();
for (int i=0, i< k-2, i++)
temp = temp → next;
free (temp);
}
int main (){
int n, x, k;
head = null;
Print f ("Enter the position for and inserting;");
Scan f ("%.d" & n);
Scan f ("%.d" & x);
Insert (x, n);
Print f ("Enter the position to delite);
Scan f ("%.d" & k);
Delete ();
Print (x);
```

return ;

}

(2)

```
#include <stdio.h>
# include <stdio.h>
Stuct node {
    int data ;
  Stuct node * next ;
}
  void print List ( Stuct node * head )
{
  print f ("%.d →" (ptr → data ));
   ptr = ptr → next; }
   print f ("Null/n");
   }
   Void push (stuct node * head , int data )
{
  Stuct node * new = (Stuct node *) mall oc
          (size of (struct node ));

     new → data = data ;
     new → next = * head ;
     * head = new ;
}
  Stauct node * merge (Stuct node *a , Stuct node* b )
{
  Stuct node fake;
```

```
struct node* tail = fake;
fake.next = NULL;
while (1){
if (a==NULL)
{
tail->next=b;
break;
}
else if (b==null)
{
tail->next=a;
break;
}
else
{
tail->next=a;
tail=a
a=a->next;
tail->next=b;
}
}
return fake.next;
}
void main()
{
int keys[]={1,2,3,4,5,6,7}
int n = size of (keys)/size of key[0]
struct node* a=NULL;*b=null;
for(int i=n-2, i>=0, i=i-2)
```

```
push(&b, key[i]);
struct node* head = merge(a,b);
printlist(head);
}
```

(3)

```
#include<stdio.h>
void find(int arr[], int a, int k){
  int total=0
  int x=0, y=0;
for(x=0, x<a, x++){
while(sum<k, y<a)
      = arr[y];
    y++;
for(x=0; x<a; x++){
  while(total<k; && y<a)
     total = arr[y]
       y++;
  if(total==0)
  {
    printf("find");
     return;}
     total -= arr[x];
  }
}
  main(void){
```

```
int arr[ ] = {9, 10, 12, 4, 1, 2, 3};
    int k = 565;
    int a = sizeof(arr) / sizeof(arr[0]);
    find(arr, a, k);
    return 0;
}
```

(4)

i) Reverse order ii) Alternate order

```
#include <stdio.h>
#define size 20
  void insert(int);
  void delete();
int queue[20], a = -1, b = -1;
  void main(){
int num; choice;
while(1){
print+("\n" new" \n");
print("1. insert \n2. Delete \n3. Print
      n4. Reverse \n4. alternate \n5. exit);
print+("\n" Enter your choice");
Scant("%d", & choice");
case 1: print+("Enter the num' to insert");
    Scant("%d", & num);
    insert(num);
    break;
```

```
printf("Reverse queue");
for(int i=size; i>0; i--)
if (queue[i]==0)
continue;
printf("%d", queue[i]);
}
break;
```

Case 3:-

```
printf("Alternate elements");
for(int i=0, i<size, i>0, i++2)
{
if (queue[i]==0)
continue;
printf("%d", queue(i));
}
break;
return 0;
}
```

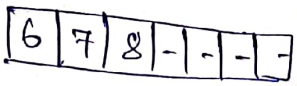
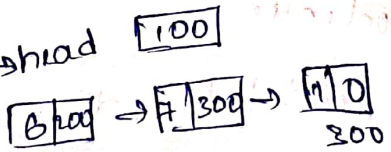
(5)

i) Arrays Vs Linked Lists

1) Both are the data structures Both are used to store the data.

2) Cost of accesing the elements.

③ Memory Requirement and utilization

Array	Linked list
Ineffective in memory utilization.	⇒ It is in dynamic size
Ex: 	
it takes at constant time	$8 \times 3 = 24$ bytes
$8 \times 4 = 32$ bytes	⇒ more requirement
⇒ Require memory in less.	

④ cost of insertion and cost of deletion

Array	Linked list
Begining - O(n)	O(1)
At end - O(1)	O(n)
ith position - O(n)	O(n)

⑤ Easy use and operations

Array	Linked list
wiser to use	⇒ less easier
linear and binary	linear

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

```
{ if (x[i])
{ xy++, i++;
}else
{ break;
}
}
return xy;
}
void change list (int x[], int a[])
{ for (int i= len[x)-1; i>=0, i--)
{
x [i+1] = x[i];
}
x[0] = a[a];
printf ("\n elements of odd array: \n")
for (int i=0; i<len(x); i++)
{
printf ("%d", x(i));
}
for (int i=0, i<len(y); i++)
{
y[i] = y(i+1]; }
Print f ("\n element of new array: \n")
for (int i=0; i< len (a); i++)
```

```
{
    Print + ("%d", a[i]);
}
    int main()
{
    int x[10] = {1,2,3}, a[10] = {4,5,6});
        change list = (a,b);
    }
```