①

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

struct node {
int value;
struct node * next;
};
void insert();
void display();
void delete();
int count();

typedef struct node DATA_NODE;

DATA_NODE* head_node, *first_node, *temp_node = 0, *prev_node,
                                                  *next_node;

int data;

int main() {
int option = 0;
print("Singly Linked List Example - All Operations\n");

while (option < 5)
{
```

```
printf ("\n Options \n");
printf ("1: Insert into Linked List \n");
printf ("2: Delete from Linked List \n");
printf ("3: Display Linked List \n");
printf ("4: count Linked List \n");
printf ("Others: Exit() \n");
printf ("Enter your option:");
scanf ("%d", &option);
switch (option)
{
  case 1:
    insert();
    break;
  case 2:
    delete();
    break;
  case 3:
    display();
    break;
  case 4:
    count(),
    break;
```

```
default :
    break;
  }
}

return 0;
}
void insert()
  {
printf ("\n Enter Element for insert Linked List : \n");
scanf (" %d", &data);
 temp_node = (DATA_NODE *) malloc (sizeof (DATA_NODE));

  temp_node -> value = data;

  if (first_node  == 0)
   {
     first_node = temp_node ;

   }
   else
   {
    head_node ->next = temp_node ;
   }
     temp_node -> next = ~~temp_node~~ 0;
     head_node = temp_node;
```

```
fflush(stdin);
}
void delete()
{
int countvalue, pos, i=0;
countvalue = count();
temp_node = first_node;
printf("\nDisplay Linked List :\n");
printf("\n Enter Position for Delete Element :\n");
scanf("%d", &pos);
if (pos >= 1 && pos <= countvalue) {
if (pos == 1)
{
temp_node = temp_node->next;
first_node = temp_node;
printf("\nDeleted Successfully \n\n");
}
else
{
while (temp_node != 0)
{
if (i == pos - 1))
{
```

```
temp_node = first_node ;
printf ("\n Display Linked Lists : \n");
while (temp_node != 0)
{
printf ("# %d # ", temp_node -> value);
count ++ ;
temp_node = temp_node -> next ;
}
printf ("\n No Of Items In Linked List : %d\n ", count),
}
int count()
{
int count =0 ;
temp_node = first_node ;
while (temp_node != 0)
{
count ++ ;
temp_node = temp_node -> next ;
}
printf ("\n No Of Items In Linked List : %d\n", count);
return count ;
}
```

(2)

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

struct Node
{
  int data;
  struct Node* next;
};
void printlist (struct Node* head)
{
    struct Node* ptr = head;
    while (ptr)
    {
printf("%d->", ptr->data);
   ptr = ptr->next;
}
  printf("NULL\n");
}
void push ( struct Node** head, int data)
{
   struct Node* newNode = (struct Node*) malloc (size of
                                    (struct Node));
```

```
newNode -> data = data;
newNode -> next = *head;
*head = newNode;
}
struct Node* ShuffleMerge(struct Node* a, struct Node* b)
{
  struct Node dummy;
  struct Node* tail = &dummy;
  dummy.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;
        tail = b;
        b = b->next;
    }
}
return dummy.next;

}

int main(void)
{
    int keys[] = {1,2,3,4,5,6,7};
    int n = sizeof(keys)/sizeof(keys[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, keys[i]);

    printf("First List:");
    printList(a);
    printf("Second List: ");
    printList(b);
```

```
struct Node * head = ShuffleMerge (a,b);
printf("After Merge : ");
printlist (head);
return 0;
}
```

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 the 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 num 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;
break;
}
push(t);
}
if(f!=1)
printf("the elements in the stack dont add up to the sum");
}
void push(int x)
{
if(top==99)
{
printf("\n stack is FULL!!!\n");
```

```
return 0;
}
x = stack[top];
top = top - 1;
return x;
}
```

④

```
#include <stdio.h>
#define SIZE 10
void insert(int);
void delete();
int queue[10], f=-1, r=-1;
void main()
{
int value, choice;
while(1)
{
printf("\n\n*** MENU ***\n");
printf("1. Insertion\n 2. Deletion\n 3. Print Reverse\n 4. Print Alternate\n 5. Exit");
printf("\nEnter your choice: ");
scanf("%d", &choice);
switch(choice) {
```

```
case 1: printf("Enter the value to be insert: ");
 scanf ("%d", &value);
  insert(value);
  break;
 case 2 : delete();
   break;
  case 3 :
       printf ("The Reversed queue is : ");
         for (int i = SIZE ; i >= 0; i--)
         {
         if (queue[i] == 0)
       continue;
       printf ("%d", queue[i]);
    }
      break;
   case 4 :
        printf ("Alternate elements of the queue are :");
          for (int i = 0; i < SIZE ; i += 2)
          {
            if (queue[i] == 0)
            continue;
```

```
        print ("%d", queue[i]);
    }
        break;

case 5: exit(0);
default: printf("\nWrong selection!!! Try again!!!");
        }
    }
}
void insert (int value)
{
    if((f=0 && r==SIZE-1) || f==r+1)
        printf("\nQueue is full!!! Insertion is not
                                        possible!!!");
    else {
        if (f==-1)
        f=0;
        r=(r+1)%SIZE;
            queue[r]=value;
        printf("\nInsertion success!!!");
    }
}
```

```
void delete()
{
    if (f == -1)
    printf("\n Queue is Empty!!! Deletion is not possible!!!");

    else
    {
    printf("\n Deleted : %d", queue[f]);

        f = (f+1) % SIZE;
        if (f == r)
        f = r = -1;
        }
    }
```

(5) i) Difference between Array and Linked List regards to their structure. Arrays are index based data structure where each element associated with an index. On the other hand, Linked list relies on references where each node consists of the data and the references to the previous and next element.

(5) ii)

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

struct Node
{
  int data;
  struct Node* next;
};
void printList(struct Node* head)
{
  struct Node* ptr = head;
  while (ptr)
  {
    printf("%d -> ", ptr->data);
    ptr = ptr->next;
  }
  printf("NULL\n");
}
void push(struct Node** head, int data)
{
  struct Node* newNode = (struct Node*)malloc
                         (sizeof(struct Node));
```

```
newNode->data = data;
newNode->next = *head;
  *head = newNode;
}
void MoveNode(struct Node** destRef, struct Node**
                                        sourceRef)
{
  if (*sourceRef == NULL)
    return;
  struct Node* newNode = *sourceRef;
  *sourceRef = (*sourceRef)->next;
newNode->next = *destRef;
  *destRef = newNode;
}
int main(void)
{
  int keys[] = {1,2,3};
  int n = sizeof(keys)/sizeof(keys[0]);
```

```
struct Node * a = NULL ;
    for (int i = n-1 ; i >= 0 ; i--)
        push(&a, keep[i]);

struct Node * b = NULL;
    for (int i = 0; i < n ; i++)
        push (&b, 2 * keep[i]);

    MoveNode (&a, &b) ;

    printf ("First List : ");
     printList (a) ;

    printf (" Second List : ") ;
     printList (b);

      return 0 ;
     }
```