DSA

Assignment - 4

Ch. Sushma
AP19110010345
CSE-F

(4) (ii)

```
#include<stdio.h>
#include<stdlib.h>
Struct node
{
  int a;
  Struct node *next; };
Void generate (struct node**);
Void display (struct node*);
Void delete (struct node**);
int main(){
Struct node *head =NULL;
generate (&head);
Printf ("\n Displaying the alternate nodes \n");
display (head);
delete( &head);
return 0; }
Void display (struct node *head) {
Static flag =0;
 if (head != NULL)
 {
   if (!(flag % 2)) {
     Printf ("%d", head -> a); }
   flag++;
   display (head -> Next); }
}
Void generate (struct node **head) {
int num, i;
```

```
struct node *temp;
printf ("Enter lenght of list : ");
scanf ("%d", &num);
for (i=num; i>0; i--) {
  temp = (struct node *) malloc (sizeof (struct node));
  temp->a = i;
  if (*head == NULL) {
    *head = temp;
    (*head) -> Next = NULL; }
  else
  {
    temp -> next = *head;
    *head = temp; } }
  }
Void delete (struct node **head) {
  struct node *temp;
  while (*head != NULL) {
    temp = *head;
    while (*head != NOLL) {
      temp = *head;
      *head = (*head) -> next;
      free (temp); }
  }
```

Output :-

```
Enter lenght of list : 10
Displaying the alternate nodes
    1 3 5 7 9
```

(4) (i)

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

```
z = frontanddelete (q), s;
Push (z, s); }
q -> front = -1;
q -> rear = -1;
for (i=0; i < capacity; i++) {
    y = topandpop (s);
    enqueque (q, y); }
print ("\n Reversed Content are : ");
display (q);
break;
Case 3:
  exit (0); } }
}
```

Wrong ✗

Output: Reversed content : 5 6 7 8 9

---

(5) (i) The difference b/w array and linked list, array are index based data structure where each element associated with an index; where as linked list relies on references where each node consists of the data and the reference to previous and next element.

(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 (size of
                                                  (struct node));
  new_node -> data = new_data;
  new_node -> next = (&head_ref);
  (*head_ref) = new_node; }
```

```
Push (&q, 4);
Push (&q, 5);
Push (&q, 6);
Printf ("Second linked list : \n");
Print list (q);
merge (p &q);
Print f ("merge Linked modified first, second. linked list = \n");
 Print list (p);
 return 0;
}
```

```
void print list (struct node * head) {
  struct node * temp = head;
  while (temp != NULL)
  {
    printf ("%d", temp → data);
      temp = temp → next;
   }
    printf ("\n");
  }
```

(2)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    Struct Node *next; };
Void push ( Struct node ** head_ref , int new_data ) {
    Struct nocle * new_node = (struct node *) malloc (size of (Struct node));
    new_node -> data = new_data;
    new_node -> next = (*head_ref) ;
    (*head_ref) = new_node; }
Void printlist (struct node * head) {
    Struct Node *temp = head;
    while (temp != NULL) {
    printf ("%d ", temp -> data );
        temp = temp -> next; }
    print f ("\n"); }
    Void merge (struct node *p, struct node **q) {
    Struct Node *p_curr = p, *q_curr = *q;
    Struct Node *p_next, *q_next;
while (p_curr != NULL && q_curr != NULL) {
    p_next = p_curr -> next;
    q_next = q_curr -> next;
    q_curr -> next = p_next;
    q_curr -> next = q.curr;
    p_curr = p_next ;
    q_curr = q_next ; }
    * q = q_curr;
int merge;
    Struct Node * p = NULL, * q = NULL;
```

```
Push (&p, 1);
Push (&p, 2);
Push (&p, 3);
Printf ("First linked list : \n");
PrintfList (p);
Push (&q, 4);
push (&q, 5);
push (&q, 6);
Printf (" Second linked list : \n");
Printf list (p)
merge (p, &q);
Printf ("Modified first linked list : \n");
print list(p,2);
return 0;
}
```

1,2

Output :-

Modified linked list {1, 2, 3, 4, 5, 6}

---

(1)

```
#include<stdio.h>
#include <stdlib.h>
struct node {
struct node *next;
};
struct node *curr, *temp;
void input (struct node)
Void delete (struct node)
void main (void) {
struct node *s;
int n;
s=NULL
```

```
do {
printf ("1.Enter the element to insert; \n");
printf ("2. Delete \n");
printf ("3. Exit \n");
printf ("Enter the choice: ");
scanf ("%d", &n);
switch (n) {
case 1: input (5);
        break;
case 3: delete (3);
        break;
} while (n != 3)
}
void input (struct node *;) {
int pos, c=1
curr 2;
printf (" Enter the element to be inserted : ");
scanf ("%d", &pos);
while (curr -> next != NULL) {
c++;
if (c == pos) {
temp = (struct node *) malloc (size of (struct node));
printf ("Enter the numbers ")
scanf ("%d", &temp -> n);
temp -> next = curr -> next
Curr -> next = temp
break; } } }
S = NULL
```

```
Void delete (struct node * z ) {
int pos, c = 1;
 Curr = z ;
 printf (" Enter the element to be deleted: ");
 Scanf ("%d", &pos);
 while (curr -> next != NULL) {
   c++;
   if (c == pos) {
   temp = curr -> next;
   Curr -> next = curr -> next -> next;
    free (temp) }
    Curr = Curr -> next; }
   Void merge (struct node * p, struct node * q ) {
     struct node * p_curr = p, *q_curr = *q;
     Struct node * p_next, * q_next;
     while (p_curr = NULL && q_curr != NULL ) {
        p_next = p_curr -> next;
        q_next = q_curr -> next;
        q_curr -> next = p_next;
        p_curr -> next = q_curr;
        p_curr = p_next;
        q_curr = q_next; }
        * q = q_curr }
     int main ( ) {
      struct node *p = NULL, * q = NULL;
        Push (& p, 1);
        Push (& p, 2);
        push (&p, 3);
        printf (" first linked list: \n");
         Print List (p);
```

```
Push (&q, 4);
Push (&q, 5);
Push (&q, 6);
Printf (modified 1st, 2nd LinkedList : \n");
 print list (p, q);
 return 0;
}
```

---

(3)

```
#include <stdio.h>
int s1[10], top1 = -1, s2[10], top2 = -1;
int s1empty() {
 if (top1 == -1)
    return 1;
 else {
    return 0; }
}
int s1top () {
 return s1[top1]; }
 int s1 pop() {
    top1--; }
int s1 push (int x) {
  s1 [++top1] = x;
} int s1 empty ()
{
  if (top2 == -1)
     return 1;
  else return 0; }
int s2 top()
{ top2--; }
int s2 push (int x) {
 s2 [++top2] = x; }
int sum (int k) {
int x;
while (s1 empty () != 1)
{
 x = s1 top ();
    s1 pop();
while (s1 empty() != 1) {
if (x + s1 top () = k)
{
 printf ("%d %d", x, s1, top());
}
s2 push (s1 top());
  s1 pop()
}
while (s2 empty () != 1) s
```

```
        s[++top] = stack[top];
        stack[top--]; }
    }}
int main()
{
  int n, i, e, k;
  printf("enter the no of elements of stack: \n");
  scanf("%d", &n);
  for(i=0; i<n; i++) {
     scanf("%d", &e);
        push(e); }
  Printf("enter the value of Constant Sum: \n");
  scanf("%d", &k);
  printf("The combinations whose sum is equal to k is: \n");
  Sum(k);
}
```