Assignment-4

1) program to insert and delete an element at n^{th} & k^{th} position in linked list.

K. Vijay rama Krishna Reddy
AP19 1100 10368
CSE-F

```
#include <stdio.h>
#include <stdlib.h>
struct linked_list
{
   int number;
   struct linked_list *next;
};
typed of struct linked_list node;
node *head = NULL, *last = NULL;
void create_linked_list();
void print_linked_list();
Void insert_at_last (int value);
voit insert_at_first (int value);
void
   insert_after(int key, int value);
```

```
void delete_item (int value);
void search_item (int value);
int main ()
{
  int key, value;
  // create a linked list
  printf ("create linked list \n");
  create_linked_list();
  print_linked_list();
// Insert value at last position to existing linked list.
  printf ("in insert new item at last \n");
  scanf ("%d", &value);
  insert_at_last (value);
  print_linked_list();
// Insert value at First position to existing linked list.
```

```
printf ("\n Insert new item at first \n");
scanf ("%d", &value);
insert_at_first (value);
print_linked_list ();
//Insert value after a defined value
printf ("\n Enter a KEY (existing item of list),
                after that you want to insert a value\n);
scanf ("%d", &key);
printf ("\n Insert new item after %d KEY \n", key);
scanf ("%d", &value);
insert_after (key, value);
Print_linked_list ();
// search an item from linked list.
Printf ("/n Enter an item to search it from list \n");
Scanf ("%d", &value);
```

```
search_item(value);

// delete value from linked list
printf("\n Enter a value, which you want to delete \n");
scanf("%d", &value);
delete_item(value);
print_linked list();

    return 0;
}

/*
  user defined functions
*/

void create_linked_list()
{
  int val;
```

```
while (1)
{
    printf ("Input a number. (Enter -1 to exit) \n");
    scanf ("%d", &value);
    if (val == -1)
        break;
    insert_at_last(val);
}
}
void insert_at_last (int value)
{
  node *temp_node;
  temp_node = (node*) malloc (sizeof(node));
```

```
temp_node -> number = value;
temp_node -> next = NULL;
// For the 1st Element
if (head == NULL)
  {
    head = temp_node;
    last = temp_node;
  }
  else
  {
    last -> next = temp_node;
    last = temp_node;
  }
}
```

```
void insert_at_ first (int value)
{
  node *temp_ node =  (node*) .malloc (size of (node));
  temp_ node -> number = value;
  temp_node -> next = head;
  head = temp_ node;
}
void insert_after (int key, int value)
{
  node * my node = head;
  int flag = 0;
  while (my node != NULL)
  {
   if (my node -> number == key)
```

```
{
    node *newNode = (node*) malloc (sizeof (node));
    newNode -> number = value;
    newNode -> next = myNode -> next;
    myNode -> next = newNode;
    printf ("%d is inserted after %d\n", value, key);
    flag = 1;
    break;
  }
  else
     myNode= myNode -> next;
}
if (flag ==0)
   printf ("key not founded !\n");
```

```
}

void delete_item (int value)

{

 node *myNode = head, *previous= NULL;

 int flag = 0;

 while (myNode != NULL)

  {

  if (myNode -> number == value)

  {

    if (previous == NULL)

      head = myNode -> next,

    else

      previous -> next = myNode -> next;

  printf("%d is deleted from list\n", value);

  flag = 1;
```

```
        free(my node);
        break;
    }
    previous = my node;
    my node = my node -> next;
}
if (flag == 0)
    printf ("key not found!\n");
}

void print_linked_list()
{
  Print ("\n your full linked list is \n");
  node * my list;
  my list = head;
```

```
while (my list != NULL)
{
printf ("%d", mylist -> number);
my list = my list -> next;
}
Puts ("!");
}
```

1st output:-

```
create Linked list
Inpat a number. (Enter -1 to exit)
1
Input a number. (Enter -1 to exit)
2
Input a number. (Enter -1 to exit)
3
```

Input a number. (Enter -1 to exit)

4

Input a number. (Enter -1 to exit)

5

Input a number. (Enter -1 to exit)

-1

your full linked list is

12345

Insert new item at last

1 2 3 4 5 6

Insert new item at first

0

your full linked list is

0 1 2 3 4 5 6

Enter a key (existing item of list), after that you want to insert a value

6

Insert a new item after 6 key

7

7 is inserted after 6

Your full linked list is

0 1 2 3 4 5 6 7

Enter an item to search it from List

3

3 is present in list. Memory address is 12348688

Enter a value, which you want to delete from List

5

5 is deleted from List

Your full linked list is

0 1 2 3 4 6 7

2) Construct a new linked list by merging alternate nodes of two lists.

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

// Data structure to store a linked list.
    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");

// Insert new node in begining

void push (struct Node** head, int data)

{
    struct Node* newNode = (struct Node*) malloc (sizeof
                                (struct Node));
    new Node -> data = data;
    new node -> next = *head;
    *head = newNode;

}

// Function to construct a linked list by merging
alternate Nodes
```

// two given linked lists using connect node

```
struct Node* shuffle Merge (struct Node* a, struct Node* b)
{
  struct node connect;
  struct node* tail = &connect;
  connect.next = NULL;
  while (1)
  {
    //empty list cases
    if (a == NULL)
    {
      tail -> Next = b;
      break;
    }
  }
  else if (b == NULL)
```

```
{
  tail -> next = a;
  break;
 }
// move two nodes to tail
else
{
  tail -> next = a;
  tail = a;
  a = a -> next;
 tail -> next = b;
 tail = b;
 b = b -> next;
 }
}
return connect.next;
```

```
}
int main (void)
{
  int keys [] = {1, 2, 3, 4, 5, 6, 7};
  int n= size of (keys) / size of 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: ");
  print List (a);
  printf ("second List: ");
  Printlist (b);
```

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

Output: First List : 1 -> 3 -> 5 -> 7 -> NULL

Second List : 2 -> 4 -> 6 -> NULL

After Merge : 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

3)

```
# include <stdio.h>

int stack (100), choice, n, top, x, i;

void push(void);

void display (void);

int main()

{

  top = -1;

  printf ("\n enter the size of stack : ");

  scanf ("%d", &n);

  printf("\n\t STACK OPERATIONS USING ARRAY");

  printf ("\n\t 1. PUSH \n\t 2. DISPLAY\n\t 3. SUB ARRAY
                                    \n\t 4. EXIT);

  do

{

  printf ("\nEnter the choice :");
```

```
scanf ("%d", &choice);
switch (choice)
{ case 1:
  { push()
    break;
  }
  case 2:
  { display()
    break;
  }
  case 3:
  { SubArraySum()
    break;
  }
case 4:
{
printf("\nIt EXIT POINT");
```

```
            break;
        }
        default:
      {
        printf ("\n\t please Enter a valid choice
                                    (1/2/3/4)");
        }
    }
}
 while (choice!=4)
    return 0;
}
void push()
  {
    if (top>=n-1)
    {
      printf("\n\t STACK is over flow");
```

```
}
else
}
{
 printf ("Enter a value to be pushed");
 scanf ("%d", &x);
 top++
 stack (top):x;
 }
}

void display ()
{
 if (top>=0)
 {
  printf("\n the elements in STACK\n");
 for(i= top; i>=0; i--)
 printf("\n %d", stack(i));
```

```
printf("\n press next choice");
}
else
{
printf("\n int STACK is empty");
}
}
int subArraySum(int stack), int sum).
{
int curr_sum, i, j;
scanf("%d", &sum);
for (i=0; i<n; i++)
{
curr_sum = stack(i) + stack[j];
// try all sub arrays starting with i
for (j= i+1; j<=n; j++)
```

```
        {
        if (curr_sum == sum)
      {
        printf ("Sum found %d and %d, i, stack[i], stack
                                                    [j];
        return 1;
      }
    if (curr_sum > sum || j == n)
        break;
  curr_sum = curr_sum + stack[j];
    }
}
 printf ("No subarray found");
  return 0;
}
int main()
}
```

```
int sum = 23;
SubArraySum(stack, n, sum);
return 0;
}
```

Output :-

```
1. PUSH
2. DISPLAY
3. SUBARRAY
4. EXIT
Enter choice : 1
Enter a value to be pushed:
1
Enter choice : 1
Enter a value to be pushed:
2
Enter choice : 1
```

Enter a value to be pushed:

3

Enter choice: 1

Enter a value to be pushed:

4

Enter choice: 2

the elements in the stack

1

2

3

4

press next choice 3

3

Sum found · 1, 2

4) implement of queue in Reverse order

a)

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

# define Max 20

void show int stack [], int size, int top)
{
 int i;
for (i=0; i< size; i++)
{
  printf ("\n value at %d is %d, top, stack (top));
    top = top-1;
  }
}
void reverse(int stack[],int qu[], int *t, int *r, int*f)
{
*f=0;
```

I used stack to reverse queue

```
while (*t>-1
{
  *r = *r+1;
  qu[*r]:Stack[*t];
  *t = *t-1;
}
while (*f<=*r)
{
  *t=*t+1;
  stack[*t] = qu[*f];
  *f = *f+1;
}
}
int main()
{
int size
intitem ,b, i; stack[MAX], queeMAX];
```

```
int top = -1, front = -1, rear = -1;
printf ("Enter size of stack");
scanf ("%d", &size);
for (i=0; i<size; i++)
{
 top = top+1;
printf ("Enter value of for position %d :: ", top);
scanf ("%d", &item);
stack(top) = item;
}
show (stack, size, top);
reverse (stack, que, &top, &rear, &front);
printf ("\n After reverse ....");
show stack, size, top);
getch ();
}
```

output:

Enter size of stack : 5

Enter value of for position 0 : : 1

" " " " " 1 : : 2

" " " " " 2 : : 3

" " " " " 3 : : 4

" " " " " 4 : : 5

value at 4 is 5

value at 3 is 4

value at 2 is 3

value at 1 is 2

value at 0 is 1

After Reverse ....

Value at 4 is 1

value at 3 is 2

value at 2 is 3

value at 1 is 4

value at 0 is 5

4(ii) program to print elements in a queue in alternate order

```
#include <stdio.h>
# define MAX 50
void insert ();
void alternate ();
void display ();
int queue_array[MAX];
int rear = -1;
int front = -1, size;
scanf ("%d", &size);
main()
{
int choice;
while(1)
{printf("1. Insert element to queue\n");
```

```
printf("2. Display element from queue\n");
printf("3. Alternate elements");
printf("4. quit\n");
printf("Enter your choice:");
scanf("%d", &choice);
switch(choice)
{
  case 1:
    insert();
      break;
case 2:
display();
  break;
  case3;
alternate();
break;
```

```
    case 4:
  exit(1);
   default:
    printf("wrong choice\n");
    }
   }
}
void insert()
{
  int add_item;
  if (rear == MAX-1)
  printf("queue overflow\n");
  else
  {
    if (front == -1)
     front = 0;
     printf("Insert the element in queue");
```

```
        scanf("%d", &odd_item);
        rear = rear + 1;
        queue_array[rear] = odd_item;
    }
}
void display()
{
  int i;
  if (front == -1)
    printf("queue is empty\n");
  else
  {
    printf("queue is :\n");
    for (i=front; i<=rear; i++)
      printf("%d", queue_array[i]);
  printf("\n");
```

```
    }
  }
void alternate ()
{
  int i, J, temp;
  printf("alternate elements are\n");
  for(i=0; i<size; i+=2)
        printf("%d\n", queue_array(i));
}
```

output:

Enter choice: 1

Enter the element in queue: 10

Enter choice: 1

Insert the element in queue: 20

Enter choice: 1

Insert the element in queue: 30

Enter choice : 1

Insert the element in queue : 40

Enter choice : 1

Insert the eement in queue : 50

Enter choice : 2

10

20

30

40

50

Enter choice : 3

10

30

50

Enter choice: 4

Exit

5)(i) How array is different from linked list?

Array

1) An array is a collection of elements of a similar data type

2) Array elements can be accessed randomly using the array index

3) Data elements are stored in contiguous locations in memory

1) Linked lists is an ordered collection of elements of same type in which each element is connected to next using pointers.

2) Random accessing is not possible in linked lists. The elements will have to be accessed sequentially.

3) New elements can be stored anywhere and a reference is created for the new element using pointers.

5(iii) program to add first node of linked list to another linked list.

```
#linked
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
  int data;
  struct node* next;
};
  void print List (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;
}
// Function take the node from the front of source
// and move it to front of destination
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--)
(struct Node));    Push(&a, keys[i]);
```

// construct 1st linked list

// construct 2nd linked list

```
struct Node* b = NULL;
```

```
for (int i=0; i<n; i++)
    push(&b, 7*keys[i]);

// move front node of 'b' and move it to the
   front of a
moveNode(&a, &b);

printf("First List:");

printList(a);

printf("Second List:");

printList(b);

return 0;
}
```

Output:

First List: b -> 1 -> 2 -> 3 -> null

Second List: 4 -> 2 -> null.