DSA
Assignment

Tummapudi Girishma
AP19110010017
CSE-F (G)

① Program to insert and delet an element at n^{th} & $*^{th}$ position in linked list.

```
#include <stdio.h>
#include <stdio.h>
struct  lincked_list
{
   int number;
   struct linked_list *next;
};
typedef struct linked_list node;
  node * head = Null, * last = NULL;
 void create_linked_list ();
 void print_linked_list ();
 void insert_at_last (int value);
 void insert_at_first (int value);
 void insert_after (int key, int value);
 void delete_item (int value);
 void search_iteam (int valeee);
int main()
{
 int key value;
 // create a linked list
 printf("create linked list \n");
   created_linked_list ();
   Print_linked_list ();
```

Scanned with CamScanner

```
// Insert value in last position of the list
printf("\n Insert at last \n");
 scanf("%d" &value);
  insert_at_last(value);
  Print_linked_list();
 // Now Insert value at 1st position
printf("\n Insest a value for 1st position \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_linKet_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 your wants
                                    to delect \n");
  scanf("%d", value);
  delect_item(value);
  print: linked_list()

  return 0;
```

Scanned with CamScanner

```
}

/*
  user defined functions
*/
void create linked list()

{ int val;
 while(1)
 { printf("Input a number. (infer -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;
     }
}
```

Scanned with CamScanner

```
void insert_at_first (int value)
{
  node * temp_node = (node *) malloc (sizeof 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)
 {int flag = 0;
   node * new Node = (node *) malloc(sizeof (node))
 newNode -> number = value ;
new Node -> next = my Node -> next ;
my Node -> next = newNode.
print (" %d is inserted after %d \n",
                                    value, key);
 flag = 1;
  break;
} else
      my. Node = myNod -> next ;
}
  if (flag == 0)
  printf("key not found ! \n");
}
  void delete_item (ints value)
 { node * my Node = head, * Previous = NULL;
     int flag = 0
```

Scanned with CamScanner

```
printf("Key not found! \n");
}
void delect_item(int value
{node *myNode = head ; * Previous = NULL;
int flag=0;
while (my Node! = NULL)
{ if (my node -> 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()
{
 printf("\n Your full linked list is \n");
  node * my list;
  mylist= head;
  while (my list! = Null)
 { Printf("%d", mylist ->number);
   mylist = mylist ->next;
  } puts("");
```

Scanned with CamScanner

}

1st output

Create a linked list
Input a number (Enter -1 to exist)
1
Input a number. (Enter -1 to exist)
1
Input a number. (Enter -1 to exist)
3
Input a number (Enter -1 to exist)
4
Input a number. (Enter -1 to exist)
5
Your full linked list is
1 2 3 4 5
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
how want to insert a value
6
Insert 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

Scanned with CamScanner

]

3 is present in list. Memory address is 1234688

Enter 0 value, which you want to delet from list

5

5 is delect from list

Your full linked list is

0 1 2 3 4 6 7 //

② Construct a newlinked list by merging alternate nodes of two lists

```
#include <studio.h>
#include <studio.h>
// Data structure to store a linked list.
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;
  } print("NULL \n");
  // Insert newnode in begining
  void Push (struct Node ** head, link data)
  { struct Node * new Node = (struct Node)* malloc
                              (size of node);
    newNode → data = data;
    newNode → next = * head;
    * head = newNode;
}
```

Scanned with CamScanner

```
// function construct a linked list by multiplying
alternative nodes
Struct Node* shuffle Merge (struct Node* a, struct Node* b)
{
   struct Node dilbar;
   struct Node *tail = &dilbar;
   dilbar.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 dibar.next;
}
int main(void)
{
   int keys[] = { 1,2,3,4,5,6,7 };
```

Scanned with CamScanner

```
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 = ");
Print list(a);
printf("second list: ");
  print list(b);
struct Node *head = shuffle Merge(a,b);
Printf("After Merge: ");
Print list(head);
return 0;
}
```

Output:

~~Print~~

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 1& stack OPERATIONS USING ARRAY");
printf("\n 1+ Push \n 2 Display \n
        3 EXIT \n 4 EXIT");
```

Scanned with CamScanner

```
do
{ printf("\n Enter the choice:");
  scanf("%d", &choice);
  switch(choice)
  { case 1:
    { push()
      break;
    }
    case 2:
    { display()
      break;
    }
    case 3;
    { Sub_Array sum()
      break;
    }
    case 4:
    { printf("\n It Exist Point");
      break;
    }
    default;
    { printf("\n It Please Enter a void
                                choice
                                (1|2|3|4)");
    }
  }
}
  while (choice != 4)
  return 0;
}
  void push()
{
```

Scanned with CamScanner

```
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 the stack is empty");
   }
}
 int subArraySum(int stack[], int sum)
 { int curr_sum, i, s;
  scanf("%d", &sum);
  for(i=0; i<n; i++)
 { curr_sum = stack[i]+stack[s];
 // by all subarrays  starting with;
```

Scanned with CamScanner

```
for (J=i+1; J<=n; J++)
{
  if (curr_sum==sum)
  {
    printf("Sum found %d and %d ", stack(i), stack(5));
    return 1;
  }
  if (curr_sum>sum || J==n)
    break;
  curr_sum = curr_sum+stack[J];
}
}
printf("\n 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 the Pushed:
(
Enter choice : 1
Enter value to be Pushed:
(

Scanned with CamScanner

```
Enter choice : 1
Enter a value to be Pushed:
2
Enter choice: 1
Enter a value to be Pushed
2
Enter a 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 stack
1
2
3
4
Press next choice
3
Sum found 1, 2
```

{ used stack to reverse queen

(4) ; implement of queen in Reverse order

```
# 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-1
```

Scanned with CamScanner

```
}
}
Void reverse (int stack[], int que[], int *t, int *r, int *f)
{
  *f = 0;
  while (*t > -1)
  { *r = *r + 1;
    que[*r] = stack[*t];
    *t = *t - 1;
  }
  while (*f <= *r)
  {
    *t = *t + 1;
    stack[*t] = que[*f];
    *f = *f + 1;
}
 int main()
{
 int size
int item, t, i, stack[Max], ques[Max];
int top = -1, fount = -1, rear = -1;
printf("Enter size of stack");
 Scanf("%d", &size);
 for (i=0; i<size, i++)
{
   top = top + 1;
   printf("Entre value of for position %d:", top);
   scanf("%d", &item);
   stack(top) = item;
}
   show(stack, size, top);
    reverse (stack, quene, &top, &rear, &fount);
```

Scanned with CamScanner

```
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 of 4 is 5
value of 3 is 4
value of 2 is 3
value of 2 is 3
value of 1 is 2
value of 0 is 1

After reverse---
value at 4 is 1
value of 3 is 2
Value of 2 is 3
Value of 1 is 4
value of 0 is 5

(4) ///Program to print elements in a queue in alternative order

```
#include <stdio.h>
#define Max 50
void insert();
void alternative();
void display();
int queue_array [MAX];
int rear = -1;
```

Scanned with CamScanner

```
int front = -1 ; size
Scanf("%.d", & size);
main()
{
  int choice;
  while(1)
  { printf("1. Insert element of queen \n");
    printf("2 Display element fromqueen\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;
      case 3;
     alternate();
     break;
     case 4;
     exit(1);
     default;
      printf("Wrong choice \n");
    }
  }
}  void insert()
{ int odd. item;
```

Scanned with CamScanner

```
if (rear == Max-1)
printf ("Queue overflow \n");
else
{
  if (front == -1)
  front = 0;
  printf ("Insert the element in Queue");
  scanf ("%d", &add_item);
  rear = rear+1;
  queue_array[rear] = add_item;
  }
}
void display()
{
int i;
if (front == -1)
  printf ("Queue it 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
Insert the element in queue : 10

Scanned with CamScanner

Enter choice: 12
Enter 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 element in queue: 50

Enter choice: 2

10
20
30
40
50

Enter choice: 4

Exit

(5) (i) How array is different from linked lists

Array	linked list
1 An array is collection of elements of a similar data type	1 linked lists is an ordered collection of elements of same type in which each element is connected to next using pointers
2 Array elements can be accessed randomly using the array index	2 Random accessing is not possible in linked lists like elements will have to be accessed sequentially
3 Data elements are stored in contiguous locations in memory	3 New elements can be stored anywhere and reference is created

Scanned with CamScanner

for the new element
using pointer

⑤ // Program to add first node of linked list to another linked list.

```
#include <stdio.h>
#include <stdio.h>
struct node
{ int data;
  struct abde *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 *new Node = (struct Node*)malloc
                          (sizeof struct
                           Node));
  newNode → data = data;
  newNode →next = *head;
   *head = newnode
}
// function ion take the node from the fount of
  Source
// and move it to fount of destination
void MoveNode (struct Node** destRef, struct Node**
                                      sourceRef)
```

Scanned with CamScanner

```
{
 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 = size of (keys) / size of keys (0);
     struct Node *a = NULL;
   for (int; = n-1; i>=0; i--) } // construct
                                 1st linked list
        Push (& a, keys[i]);
    // construct 3rd linked list
     struct Node *b = NULL;
    for (int i=0; i<n; i++)
        Push (&b, 2*keys (i));
 // move fount node of b and move it to
the fourth of a
   Move node (& a, &b);
   Printf (" first list: ");
    printfist(a);
    printf ("second list:");
    print list (b);
     return 0;
 }
```

Scanned with CamScanner

output:

first list : 6 → 1 → 2 → 3 → null

second list : 4 → 2 → null

⇒

Scanned with CamScanner