Assignment

P. Manoj Sai
AP19 1100101 09
CSE G

1) # include <stdio.h>
include <stdlib.h>
struct linked_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 number);
void insert_after(int key, int number);
void delete_item(int number);
void search_item (int number);

int main()
{
int key, number;
//* We need to create a linked list
Printf ("Create a Linked List: \n");
Create_linked_list();
Print_linked_list();
Printf ("In Insert new item at last \n");
Printf ("%d", number);
insert_at_last(number);
Print_linked_list();
Prinf("\n Insert new item at First \n");
Scanf("%d", &number);
insert_at_first(number);
Print_linked_list();
Printf("In Insert new item at first \n");

```
scanf("%d",&number);
insert_at_first(value);
Print_linked_list();
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", &number);
insert_after( key, number);
Print_linked_list();
Printf("\n Enter an item to search it from list \n");
scanf("%d", &number);
search_item( value);
// Delete value from linked list
Printf("\n Enter a value, which we want to delete\n");
scanf("%d", number);
delete_item(value);
Print_linked_list();
return 0;
}
void create_linked_list()
{
int val;
while(1)
{
Printf("Input a value. (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(nodes));
    temp_node -> number = value;
    temp_node -> next = NULL;
    if(head == NULL)
    {
        head = temp_node;
        last = temp_node;
    }
    else
    {
        last -> next = temp_node;
        last = temp_node;
    }
}
void insert_at_first(int value number)
{
    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(myNode -> number == key)
      {
        node * newnode = (node* malloc(sizeof(node));
```

```
newNode -> number= value;
new Node -> number= value;
newNode -> next = myNode -> next;
mynode -> next = newNode;
Printf("%d is inserted after %d \n", number, key);
flag=1;
break;
}
else
    myNode= myNode-> neext;
}
if (flag==0)
Printf(" key not found ! \n");
}
void delete_item( int number)
{
node* myNode=head, *Previous=NULL;
int flag=0;
while(mynode != NULL)
{
if(myNode != NULL)
{ if( Previous==NULL)
head=myNode->next;
else
Previous-> next = my Node-> next;
Printf("%d is deleted from list\n", number);
flag=1;
free(myNode);
break;
}
Previous= myNode;
myNode= myNode->next;
}
if(flag==0)
{   Printf(" Key not found !\n");
```

```
void Print_linked_list()
{
    Printf(" \n your full linked list is\ n");
     node* mylist;
     mylist= head;
     while (mylist != NULL)
     {
         Printf( "%d", mylist -> number);
         mylist=  mylist -> next;
     }
     Puts(" ");
}
```

1st output:

Input a number (Enter -1 to exit)

5

Input a number

7

Input a value

8

Input a value

9

Input a value

4

Input a value

-1

Full Linked list is

5 5 7 8 9 4

Insert a new item at last

5 7 8 9 4 3

Insert a new item at first

6

your full linked list is

6 5 7 8 9 4 3

Enter a key

3

Insert new item after 6 key

2

2 is inserted after 3

Your full linked list is

~~0 1 2 3 4 5 6 7~~ 6 5 7 8 9 4 3 2

Enter an item to search it from List

3

3 is presen in List

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~~ 6 7 8 9 4 3 2

2)
```
#include<stdio.h>
#include<stdlib.h>
Struct Node
{
    int mydata;
    Struct Node* next;
};
Void PrintList(struct Node* head)
{
    struct Node* p = head;
    while(p)
    {
        Printf("%d->", p->mydata);
        p = p->next;
    }
    PrintF("NULL\n");
    Void Push(Struct Node** head, int mydata)
    {
        struct Node* newNode = (Struct Node*)
                               malloc(sizeof(structNode));
        newNode->data = data;
        newNode->next = *head;
        *head = newNode;
    }
Struct Node* ShuffleMerge(struct Node* a, Struct Node* b)
{
    struct Node num;
    struct Node* tail = &num;
    dilbar.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 num->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:\n");
     Print List(a);
     Printf("Second List:\n");
      Print List(b);
       struct Node* head=Shoffle Merge(a,b);
       Printf("After merge:\n");
        PrintList(head);
   }  return 0;
```

2nd output

First List: 4 → 3 → 2 → 1 → null

Second List: 7 → 5 → 6 → null

After Merge: 4 → 7 → 3 → 5 → 2 → 6 → 1 → null

③

```
#include<stdio.h>
int stack[100], choice, num, top, i;
void push(void);
void display(void);
int main()
{
   top=-1;
  printf("\n Enter the size of stack:");
  scanf("%d", &num)
  printf("\n\t stack operations using Array");
  printf("\n\t 1. PUSH \n\t2.DISPLAY \n\t3.SUBARRAY
                                  \n\t 4.EXIT);
  do
  {
     printf("\n Enter ur choice:");
     scanf ("%d", &choice);
     switch(choice)
     {
        case 1:
         { push()
           break;
        case2: display()
               break;
        case3: subarray sum()
               break;
        case4: printf("\n\t EXIT POINT");
               break;
        default : printf("\n\t Enter correct option
  }
}
```

```
while (choice != 4)
  return 0;
}
void Push()
{
   if(top>=num-1)
   {
     Printf("\n\t Stack is overflow:");
   }
   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 cu_sum, i, j;
    scanf("%d", &sum);
    for(i=0; i<n; i++)
    {
        cu_sum = stack[i] * stack[5];
        for(j=i+1; j<=n; j++)
        { if( cu_sum == sum)
            { Printf("Sum found %d and %d,
                                stack[i], stack[j];
                return 1;
            }
            if( cu_sum>sum || j==n)
                break;
            cu_sum= cu_sum+stack[j];
        }
    }
    Printf(" No Sub array found");
    return 0;
}
int main()
{
    int sum=23;
    SubArraySum(stack, num, sum);
    return 0;
}
```

Output:

1. PUSH
2. DISPLAY
3. SOBARRAY
4. EXIT

Enter choice: 1

Enter a value to be Pushed: 1

Enter choice: 1

Enter a value to be Pushed: 4

Enter choice: 1

Enter a value to be Pushed: 8

Enter choice: 1

Enter a value to be Pushed: 9

Enter choice: 1

The Elements in stack.

The elements in stack

Enter a value to be Pushed: 2

Enter choice: 2

The Elements in stack

4 8

8

9

2

Press next choice 3

12

Sum found 4, 8

7)
```
// #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("In Value at %d is %d", top, stack[top]);
    top=top-1;
 }
}
void reverse (int stack[], int queue[], int *x,
                          int* y, int *z)
 {
    *z=0
    while(*x>-1)
     {
        *y=*y+1;
         queue[*y]=stack[*x];
          *x=*x+1;
     }
    while(*z<=*y)
    {
       *x=*x+1;
       stack[*x]=queue[*z];
         *z=*z+1;
    }
}
```

```
int main()
{ int size
  int item, i, stack[MAX], queue[MAX];
  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 at for Position %d :", top);
      Scanf("%d", &item);
      stack[top]=item;
   }
   show(stack, size, top);
   reverse(stack, queue, &top, &rear, &front);
   Printf("\n After Reverse ...");
    show(stack, size, top);
    getch();
}
```

Output:-

Enter size of stack: 3
Enter value at 0 : 4
1 : 8
2 : 9
value at 2 is 9
1 is 8
0 is 4

After reverse

value at 2 is 4
value at 1 is 8
value at 0 is 9

4) 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. Alternative elements");
    Printf("4. Quit \n");
    Printf("Enter ur 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 add_item;
  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 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=i+2)
      printf("%d \n", queue_array[i]);
  }
```

Output

```
Enter choice: 1
Insert the element in queue: 1
Enter ur choice: 1
Insert the element in queue: 2
Enter ur choice: 1
Insert the element in queue: 3
Enter ur choice: 1
Insert the element in queue: 4
Enter ur choice: 1
Insert the element in queue: 5
Enter ur choice: 2
1
2
3
4
5
Enter ur choice: 3
1
3
5
Enter ur choice: 4
Exit.
```

5) How array is different from Linked List.

Array	Linked List
1) Arrays are index based data structure where each element associated with an index.	1) Linked List relies on references where each node consists of the data and the references to the previous and next element
2) Array is a set of similar data objects stored in sequential memory locations under variable name	2) Linked list is a data structure which contains a sequence of the elements where each elements is linked to its next element
3) Specified during declaration	3) No need to specify; grow and shrink during execution.
4) Stored consecutively	4) Stored randomly.

ii)

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
    int mydata;
    Struct Node * next;
};
Void PrintList(Struct Node * head)
{
    Struct Node * P= head;
    while(P)
    {
        Printf("%d ->", Ptr-> mydata);
        P= P-> next;
    }
    Printf("NULL \n");
```

```
Void Push(struct node **head, int mydata)
{
  StructNode * Newnode = (struct Node*) malloc(sizeof(Struct
                                                  Node));
  newNode->data= mydata;
  newNode -> next= *head;
  *head= newNode;
}
Void MoveNode (Struct Node** destRef, StructNode**
                                          Source Ref)
{
  if( *Source Ref== NULL)
      return;
  Struct Node* newNode= *Source Ref;
  *Source Ref= (*SourceRef)-> next;
  newNode -> next= *destRef;
  *destRef= newNode;
}
int main (void)
{
  int keys[]= {1,2,3}
  int n= sizeof(keys)/sizeof[key[0];
  Struct Node* a=NULL;
  for( int i= n-1; i>=0; i--)
      Push(&a, keys[i]);
  Struct Node* b=NULL;
  for( int i=0; i<n; i++)
      Push(&b, *keys[i]);
```

```
MoveNode(&a, &b);
Printf("First List: ");
Print List(a);
Printf("second List: ");
Print List(b);
return 0;
}
```

Output:

First List : 6 → 1 → 2 → 3 → null

Second List : 4 → 2 → null