ASSIGNMENT - 4

Harshitha. Lavu
AP19110010041
CSE - G

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

→
```
# include <stdio.h>
# include <stdio.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);
Void 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("\n 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(sizeof (node));
 temp_node -> number = value;
 temp_node -> next = head;
 head = temp_node;
}
 void insert_after (int key, int value)
{
  node * myNode = head;
  int flag = 0;
  while(my node != NULL)
  {
   if (myNode -> number == key)
   {
    node * newNode = (node*) malloc (sizeof (node));
    newNode -> number = value;
    new Node -> next = mynode -> next;
    my Node -> next = newNode;
    printf("%d is inserted after %d" \n", value, key);
    flag = 1;
    break = 1;
```

```
    }
    else
        myNode = myNode -> next;
}
 if (flag==0)
    printf("key not found ! \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(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", my List → number);
     my list= mylist → next;
   }
     puts(" ");
}
```

1st output :

Create linked list.

Input a number 1

Input a number 2

Input a number 3

Input a number 4

Input a number 5

Input a number -1

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 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 inserted 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 lists.

⇒

```
# include <stdio.h>
# include <stdio.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 beggining
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 to Construct a linked list by merging alternate nodes
// two given linked lists using dilbar node
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 (h==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 dlist.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, key[i]);
   printf ("First list:");
   printlist(a);
```

```
    printf("second list:");
    printlist(b);
    struct Node *head = shuffle merge(a,b);
    print("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.) Find all the elements in the stack whose sum is equal to k.

⇒
```
#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. SUBARRAY \n\t 4. Exist);
  do
  {
  printf("\n Enter the choice:");
  scanf("%d", &choice);
  switch (choice).
```

```
{
  case 1;
  {
    push()
    break;
  }
  case 2:
  {
    display()
    break;
  }
  case 3;
  {
    subArraysum()
    break;
  }
  Case 4;
  {
    printf("\n\t 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 the STACK is empty");
    }
  }
  int sub Array sum(int stack[],int sum).
 {
   int curr_sum,i,s;
   scanf("%d",&sum);
   for(i=0;i<n;i++)
   {
     curr_sum=stack[i],stack[s];
```

```
// try all sub arrays starting with i
    for(j=i+1; j<=n; j++)
    {
      if (curr_sum == sum)
      {
        printf("sum found between %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;
  subArray sum(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 a choice : 1

Enter a value to be pushed : 3

Enter a choice : 1

Enter a value to be pushed : 4

Enter a choice : 2

The elements in stack : 1
2
3
4

press next choice : 3.

Sum found 1, 2

(4) Write a program to print elements in a queue.

(i) 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 = top-1;
}
```

```
}
void reverse (int stack[], int qu[], int *t, int *r, int *f)
{
 *f = 0;
 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
 int item, t, i, stack [MAX], qure[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 of for position %d :: ", top);
  scanf("%d", &item);
  stack[top] = item;
```

```
}
show(stack, size, top);
reverse(stack, queue, &stop, &rear, &front);
printf("\n After Reverse ----");
show(stack, size, top);
getch();
}
```

Output:

```
Enter size at stack: 5

Enter value of for position 0 :: 1

Enter value of for position 1 :: 2

Enter value of for position 2 :: 3

Enter value of for position 3 :: 4

Enter value of for position 4 :: 5

value of 4 is 5

value of 3 is 4

value of 2 is 3

value of 1 is 2

value of 0 is 1

After reverse.

value of 4 is 1

value of 3 is 2

value of 2 is 3

value of 1 is 4

value of 0 is 5.
```

(ii) 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 elemen");
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[];
    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 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 ; 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
Enter element in queue : 20
Enter choice : 1
Enter element in queue : 30
Enter choice : 1
Enter element in queue : 40
Enter choice : 1
Enter element 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	Linked List
* An array is a collection of elements of a similar data type.	* Linked lists is an ordered collection of elements of same type in which each element is connected to next using pointers.
* Array elements can be accessed randomly using the array index.	* Random accessing is not possible in linked lists the elements will have to be accessed sequentially.
* Data elements are stored in contiguous locations in memory.	* New elements can be stored anywhere a reference is created for new element using pointers.

(ii) Program to add first node of linked list to another linked list.

⇒

```
#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;
}
void movenode (struct Node** destRef, struct Node** Source Ref)
{
   if (*sourceref == NULL)
      return;
    struct node * new node = * source Ref;
   *Source Ref = (*Source Ref) → next;
    newnode →next =* destRef;
    * dest Ref = newNode;
 }
  int main (void)
  {
```

```
int keys[] = {1,2,3}
int n = size of (keys) / size of (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, 2*keys[i]);
Move node (&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