DATA STRUCTURES LAB

Day 2

Lab experiments

1 . #include<stdio.h>

#include<conio.h>

Int main()

{

```
Int a[20],n,I,num,pos;
Printf(“enter the elements:”);
Scanf(“%d”,&n);
Printf(“enter the elemnets:”);
For(i=0;i<n;i++)
{
        Printf(“\n a[%d]=”,i);
        Scanf(“%d”,&a[i]);
}
```

Printf(“enter the number to be inserted:”);

Scanf(“%d”,&num);

Printf(“enter the position to be inserted:”);

Scanf(“%d”,&pos);

For(i=n-1;i>=pos;i--)

```
            A[i+1]=a[i];
            A[pos]=num;
            N=n+1;
            Printf(“\n array elements after insertion of %d:”,num);
            For(i=0;i<n;i++)
            Printf(“\n a[%d]=%d”,I,a[i]);
Printf(“enter the position to be delete:”);
Scanf(“%d”,&pos);
For(i=pos;i<n-1;i++)
            A[i]=a[i+1];
            n--;
   printf(“\n array elements are”);
   for(i=0;i<n;i++)
   printf(“\n a[%d]=%d”,I,a[i]);
return 0;
}
```

```
2 . #include<stdio.h>
Int main()
{
```

```
        Int a[100],search,c,num;
        Printf(“enter the number of elements:”);
        Scanf(“%d”,&num);
        Printf(“enter the elements:”);
        For(c=0;c<num;c++)
                Scanf(“%d”,&a[c]);
        Printf(“enter the number to be searched:”);
        Scanf(“%d”,&search);
        For(c=0;c<num;c++)
        {
                If(a[c]==search)
                {
                        Printf(“%d is present at location %d\n”,search,c+1);
                        Break;
                }
        }
        If(c==num)
        {
                Printf(“%d is not preasent in array\n”,search);
        }
        Return 0;
}
```

3. #include<stdio.h>

#include<conio.h>

Int main()

{

Int first,last,middle,n,search,a[100],I;

Printf(“enter the number of elements:”);

Scanf(“%d”,&n);

Printf(“enter the elements:”);

For(i=0;i<n;i++)

Scanf(“%d”,&a[i]);

Printf(“enter the value to be searched:”);

Scanf(“%d”,&search);

First=0;

Last=n-1;

Middle=(first+last)/2;

While(first<=last)

{

If(a[middle]<search)

First=middle+1;

Else if(a[middle]==search)

{

Printf(“%d is found at location %d\n”,search,middle+1);

Break;

}

Else

Last=middle-1;

```
                Middle=(first+last)/2;
        }
        If(first>last)
            Printf("not found! %d is not present in list\n",search);
        Return 0;

}
```

4.
```
#include <stdio.h>
#define MAXSIZE 5

Struct stack
{
   Int stk[MAXSIZE];
   Int top;
};
Typedef struct stack STACK;
STACK s;
```

```
Void push(void);
Int  pop(void);
Void display(void);

Int main ()
{
   Int choice;
   Int option = 1;
   s.top = -1;

   printf (“STACK OPERATION\n”);
   while (option)
   {
      Printf (“----------------------------------------\n”);
      Printf (“     1   →   PUSH            \n”);
      Printf (“     2   →   POP            \n”);
      Printf (“     3   →   DISPLAY            \n”);
      Printf (“     4   →   EXIT         \n”);
      Printf (“----------------------------------------\n”);

      Printf (“Enter your choice\n”);
      Scanf    (“%d”, &choice);
      Switch (choice)
      {
      Case 1:
         Push();
         Break;
      Case 2:
         Pop();
```

```
            Break;
        Case 3:
            Display();
            Break;
        Case 4:
            Return 0;
        }
        Fflush (stdin);
        Printf (“Do you want to continue(Type 0 or 1)?\n”);
        Scanf    (“%d”, &option);
    }
}
/*  Function to add an element to the stack */
Void push ()
{
    Int num;
    If (s.top == (MAXSIZE – 1))
    {
        Printf (“Stack is Full\n”);
        Return;
    }
    Else
    {
        Printf (“Enter the element to be pushed\n”);
        Scanf (“%d”, &num);
        s.top = s.top + 1;
        s.stk[s.top] = num;
    }
    Return;
```

```
}
/*  Function to delete an element from the stack */
Int pop ()
{
   Int num;
   If (s.top == - 1)
   {
      Printf (“Stack is Empty\n”);
      Return (s.top);
   }
   Else
   {
      Num = s.stk[s.top];
      Printf (“poped element is = %dn”, s.stk[s.top]);
      s.top = s.top – 1;
   }
   Return(num);
}
/*  Function to display the status of the stack */
Void display ()
{
   Int I;
   If (s.top == -1)
   {
      Printf (“Stack is empty\n”);
      Return;
   }
   Else
   {
```

```
    Printf (“\n The status of the stack is \n”);
    For (I = s.top; I >= 0; i--)
    {
       Printf (“%d\n”, s.stk[i]);
     }
```

5.
```
#include <stdio.h>
#include <stdlib.h>
Struct Node {
  Int data;
  Struct Node* next;
};
Void insertAtBeginning(struct Node** head_ref, int new_data) {
  Struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  New_node->data = new_data;
  New_node->next = (*head_ref);
  (*head_ref) = new_node;
}
Void insertAfter(struct Node* prev_node, int new_data) {
```

```
  If (prev_node == NULL) {
  Printf("the given previous node cannot be NULL");
  Return;
  }
  Struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  New_node->data = new_data;
  New_node->next = prev_node->next;
  Prev_node->next = new_node;
}
Void insertAtEnd(struct Node** head_ref, int new_data) {
  Struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  Struct Node* last = *head_ref;
  New_node->data = new_data;
  New_node->next = NULL;
  If (*head_ref == NULL) {
  *head_ref = new_node;
  Return;
  }
  While (last->next != NULL) last = last->next;
  Last->next = new_node;
  Return;
}
Void deleteNode(struct Node** head_ref, int key) {
  Struct Node *temp = *head_ref, *prev;
  If (temp != NULL && temp->data == key) {
  *head_ref = temp->next;
  Free(temp);
  Return;
  }
```

```
  While (temp != NULL && temp->data != key) {
  Prev = temp;
  Temp = temp->next;
  }
  If (temp == NULL) return;
  Prev->next = temp->next;
  Free(temp);
}
Int searchNode(struct Node** head_ref, int key) {
  Struct Node* current = *head_ref;
  While (current != NULL) {
  If (current->data == key) return 1;
  Current = current->next;
  }
  Return 0;
}
Void sortLinkedList(struct Node** head_ref) {
  Struct Node *current = *head_ref, *index = NULL;
  Int temp;
  If (head_ref == NULL) {
  Return;
  } else {
  While (current != NULL) {
    Index = current->next;
    While (index != NULL) {
    If (current->data > index->data) {
      Temp = current->data;
      Current->data = index->data;
      Index->data = temp;
```

```
    }
    Index = index->next;
    }
    Current = current->next;
  }
  }
}
Void printList(struct Node* node) {
  While (node != NULL) {
  Printf(“ %d “, node->data);
  Node = node->next;
  }
}
Int main() {
  Struct Node* head = NULL;
  insertAtEnd(&head, 1);
  insertAtBeginning(&head, 2);
  insertAtBeginning(&head, 3);
  insertAtEnd(&head, 4);
  insertAfter(head->next, 5);
  printf(“Linked list: “);
  printList(head);
  printf(“\nAfter deleting an element: “);
  deleteNode(&head, 3);
  printList(head);
  int item_to_find = 3;
  if (searchNode(&head, item_to_find)) {
  printf(“\n%d is found”, item_to_find);
  } else {
```

```
            Printf(“\n%d is not found”, item_to_find);
  }
  sortLinkedList(&head);
  printf(“\nSorted List: “);
  printList(head);
}
```

```
6.#include<stdio.h>
#define n 5
Int main()
{
   Int queue[n],ch=1,front=0,rear=0,I,j=1,x=n;
   Printf(“Queue using Array”);
   Printf(“\n1.Insertion \n2.Deletion \n3.Display \n4.Exit”);
   While(ch)
   {-
      Printf(“\nEnter the Choice:”);
      Scanf(“%d”,&ch);
```

```
Switch(ch)
{
Case 1:
   If(rear==x)
      Printf("\n Queue is Full");
   Else
   {
      Printf("\n Enter no %d:",j++);
      Scanf("%d",&queue[rear++]);
   }
   Break;
Case 2:
   If(front==rear)
   {
      Printf("\n Queue is empty");
   }
   Else
   {
      Printf("\n Deleted Element is %d",queue[front++]);
      X++;
   }
   Break;
Case 3:
   Printf("\nQueue Elements are:\n ");
   If(front==rear)
      Printf("\n Queue is Empty");
   Else
   {
      For(i=front; i<rear; i++)
```

```
            {
               Printf(“%d”,queue[i]);
               Printf(“\n”);
            }
            Break;
         Case 4:
            Return 0;
         Default:
            Printf(“Wrong Choice: please see the options”);
         }
      }
   }
   Return 0;
}
```