Laboratory Practical Report

of

**DATA STRUCTURE AND ALGORITHM**

**(ICT ED 435)**

Submitted To

**TRIBHUVAN UNIVERSITY**

In Partial Fulfillment of the Requirements of the course

**B.Ed. ICTE 3rd Semester**

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-0214-0054-2019

Under the guidance of

**Mr. Atul Bhattarai**

Faculty Member, BICTE Program

Sukuna Multiple Campus

Sundarharaincha-12, Morang

(Internal Examiner)

Submitted for the Final Examination on: 2078/ 12/30

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(External Examiner)

***Table of Content***

[**1.** **Write a C program to read marks of ‘n’ student and display using malloc() function.** 1](#_Toc100687826)

[**2.** **Write a C program to implement stack operations.** 2](#_Toc100687827)

[**3.** **Write a C program to implement linear queue operations.** 4](#_Toc100687828)

[**4.** **Write a C program to implement singly linked list operations.** 6](#_Toc100687829)

[**5.** **Write a C program to solve the problem of TOH(Tower of Hanoi).** 9](#_Toc100687830)

[**6.** **Write a C program to print Fibonacci series using recursive function.** 10](#_Toc100687831)

[**7.** **Write a C program to implement merge sort.** 11](#_Toc100687832)

[**8.** **Write a C program to implement sequential search.** 13](#_Toc100687833)

# **Write a C program to read marks of ‘n’ student and display using malloc() function.**

#include<stdio.h>

#include<stdlib.h>

int main()

{

float \*p;

int i, n;

printf("Enter the number of students: ");

scanf("%d", &n);

p = (float\*)malloc(n\*sizeof(float));

if(p==NULL)

{

printf("Memory allocation failed");

exit(1);

}

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

{

printf("Enter marks for %d student: ", i+1);

scanf("%f", p+i);

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* Printing the marks of students \*\*\*\*\*\*\*\*\*\*\*\n");

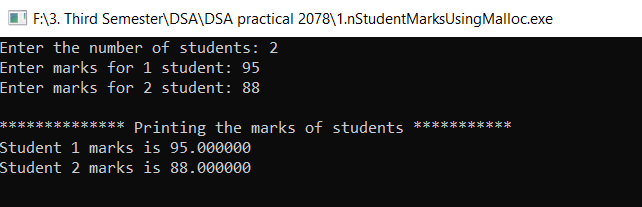
for(i=0;i<n;i++){

printf("Student %d marks is %f \n", i+1, \*(p+i));

}

return 0;

}



# **Write a C program to implement stack operations.**

#include <stdio.h>

int MAXSIZE = 8;

int stack[8];

int top = -1;

int isempty() {

if(top == -1)

return 1;

else

return 0;

}

int isfull() {

if(top == MAXSIZE)

return 1;

else

return 0;

}

int peek() {

return stack[top];

}

int pop() {

int data;

if(!isempty()) {

data = stack[top];

top = top - 1;

return data;

} else {

printf("Could not retrieve data, Stack is empty.\n");

}

}

int push(int data) {

if(!isfull()) {

top = top + 1;

stack[top] = data;

} else {

printf("Could not insert data, Stack is full.\n");

}

}

int main() {

push(3);

push(5);

push(9);

push(1);

push(12);

push(15);

printf("Element at top of the stack: %d\n" ,peek());

printf("Elements: \n");

while(!isempty()) {

int data = pop();

printf("%d\n",data);

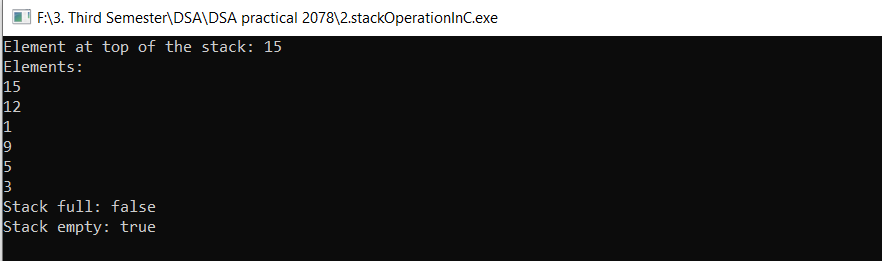
}

printf("Stack full: %s\n" , isfull()?"true":"false");

printf("Stack empty: %s\n" , isempty()?"true":"false");

return 0;

}



# **Write a C program to implement linear queue operations.**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

struct queue{

int value[MAX];

int front;

int rear;

}q;

void enQueue(int x) {

if(q.rear == MAX-1){

printf("Queue is full");

return;

}

if(q.rear == -1 && q.front == -1){

q.front = q.rear = 0;

}

else

q.rear++;

q.value[q.rear] = x;

}

int deQueue() {

int x;

if(q.front == -1 || q.front > q.rear) {

printf(" Queue is empty");

return -1;

}

x = q.value[q.front++];

return x;

}

void displayQ() {

int i;

for(i=q.front; i<= q.rear; i++)

printf("%d ", q.value[i]);

}

int main() {

int ch, p;

q.rear=q.front=-1;

while(1){

printf("\n 1-> INSERT");

printf("\n 2->DELETE ");

printf("\n 3->DISPLAY");

printf("\n 4->EXIT ");

scanf("%d",&ch);

switch(ch){

case 1: printf("\n Enter an element to insert: ");

scanf("%d",&p);

enQueue(p);

break;

case 2: p = deQueue();

if((q.front == -1 || q.front > q.rear) && p==-1)

break;

else

printf("\n The deleted element is %d",p);

break;

case 3: displayQ();

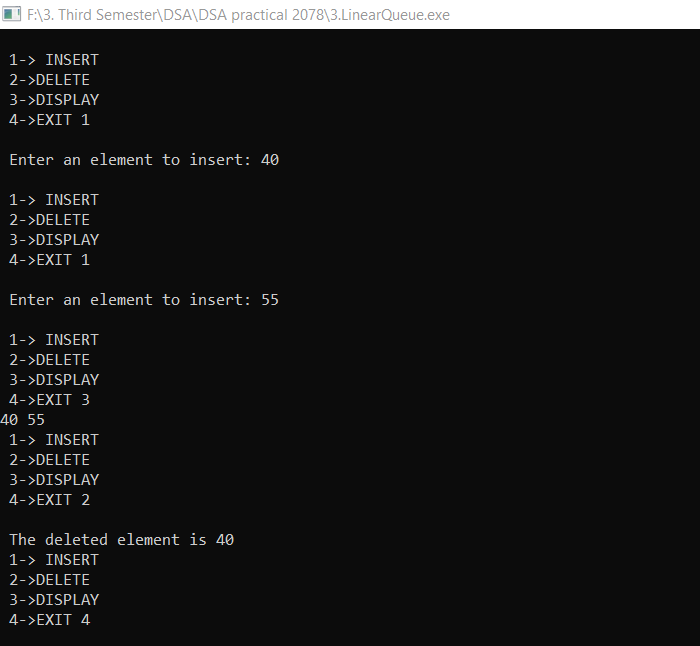
break;

case 4: exit(0);

}

}

}



# **Write a C program to implement singly linked list operations.**

#include<stdio.h>

#include<stdlib.h>

struct Node;

typedef struct Node \* PtrToNode;

typedef PtrToNode List;

typedef PtrToNode Position;

struct Node

{

int e;

Position next;

};

void Insert(int x, List l, Position p)

{

Position TmpCell;

TmpCell = (struct Node\*) malloc(sizeof(struct Node));

if(TmpCell == NULL)

printf("Memory out of space\n");

else

{

TmpCell->e = x;

TmpCell->next = p->next;

p->next = TmpCell;

}

}

int isLast(Position p)

{

return (p->next == NULL);

}

Position FindPrevious(int x, List l)

{

Position p = l;

while(p->next != NULL && p->next->e != x)

p = p->next;

return p;

}

void Delete(int x, List l)

{

Position p, TmpCell;

p = FindPrevious(x, l);

if(!isLast(p))

{

TmpCell = p->next;

p->next = TmpCell->next;

free(TmpCell);

}

else

printf("Element does not exist!!!\n");

}

void Display(List l)

{

printf("The list element are :: ");

Position p = l->next;

while(p != NULL)

{

printf("%d -> ", p->e);

p = p->next;

}

}

void Merge(List l, List l1)

{

int i, n, x, j;

Position p;

printf("Enter the number of elements to be merged :: ");

scanf("%d",&n);

for(i = 1; i <= n; i++)

{

p = l1;

scanf("%d", &x);

for(j = 1; j < i; j++)

p = p->next;

Insert(x, l1, p);

}

printf("The new List :: ");

Display(l1);

printf("The merged List ::");

p = l;

while(p->next != NULL)

{

p = p->next;

}

p->next = l1->next;

Display(l);

}

int main()

{

int x, pos, ch, i;

List l, l1;

l = (struct Node \*) malloc(sizeof(struct Node));

l->next = NULL;

List p = l;

printf("LINKED LIST IMPLEMENTATION OF LIST ADT\n\n");

do

{

printf("\n\n1. INSERT\t 2. DELETE\t 3. MERGE\t 4. PRINT\t 5. QUIT\n\nEnter the choice :: ");

scanf("%d", &ch);

switch(ch)

{

case 1:

p = l;

printf("Enter the element to be inserted :: ");

scanf("%d",&x);

printf("Enter the position of the element :: ");

scanf("%d",&pos);

for(i = 1; i < pos; i++)

{

p = p->next;

}

Insert(x,l,p);

break;

case 2:

p = l;

printf("Enter the element to be deleted :: ");

scanf("%d",&x);

Delete(x,p);

break;

case 3:

l1 = (struct Node \*) malloc(sizeof(struct Node));

l1->next = NULL;

Merge(l, l1);

break;

case 4:

Display(l);

break;

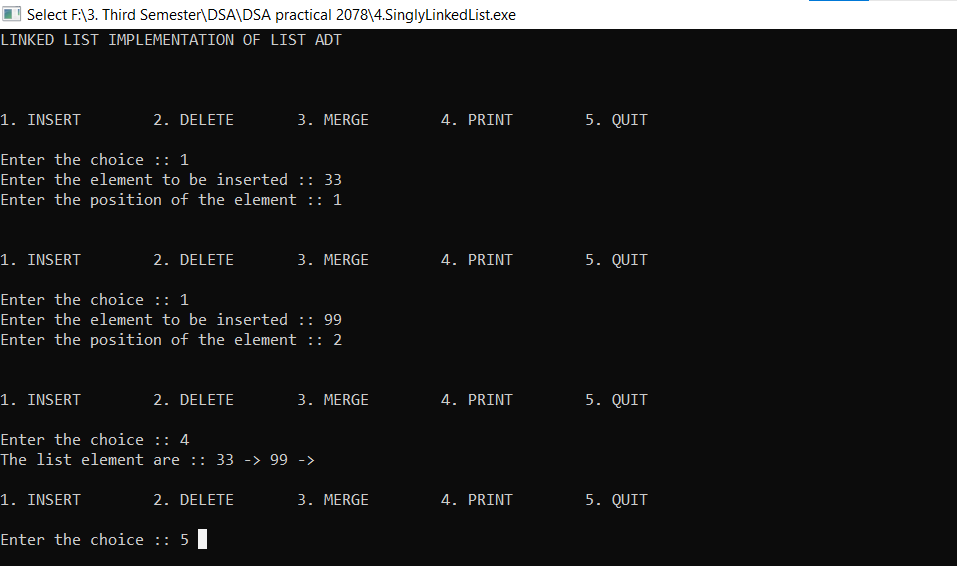
}

}

while(ch<5);

return 0;

}



# **Write a C program to solve the problem of TOH (Tower of Hanoi).**

#include <stdio.h>

void towers(int, char, char, char);

int main()

{

int num;

printf("Enter the number of disks : ");

scanf("%d", &num);

printf("The sequence of moves involved in the Tower of Hanoi are :\n");

towers(num, 'A', 'C', 'B');

return 0;

}

void towers(int num, char frompeg, char topeg, char auxpeg)

{

if (num == 1)

{

printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);

return;

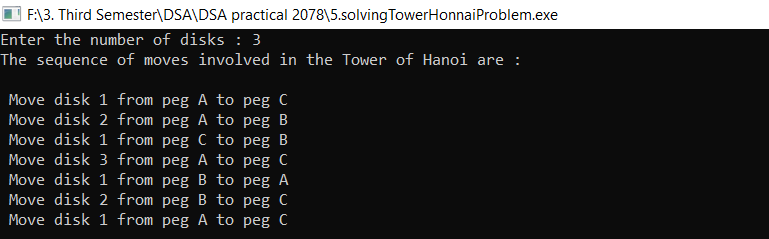
}

towers(num - 1, frompeg, auxpeg, topeg);

printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);

towers(num - 1, auxpeg, topeg, frompeg);

}



# **Write a C program to print Fibonacci series using recursive function.**

#include<stdio.h>

int fibonacci(int);

int main(void)

{

int terms;

printf("Enter terms: ");

scanf("%d", &terms);

for(int n = 0; n < terms; n++)

{

printf("%d ", fibonacci(n));

}

return 0; // return 0 to operating system

}

int fibonacci(int num)

{

//base condition

if(num == 0 || num == 1)

{

return num;

}

else

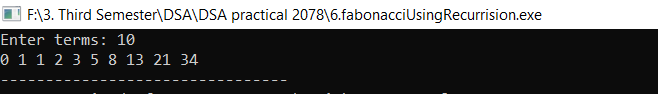
{

// recursive call

return fibonacci(num-1) + fibonacci(num-2);

}

}



# **Write a C program to implement merge sort.**

#include <stdio.h>

#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };

int b[10];

void merging(int low, int mid, int high) {

int l1, l2, i;

for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {

if(a[l1] <= a[l2])

b[i] = a[l1++];

else

b[i] = a[l2++];

}

while(l1 <= mid)

b[i++] = a[l1++];

while(l2 <= high)

b[i++] = a[l2++];

for(i = low; i <= high; i++)

a[i] = b[i];

}

void sort(int low, int high) {

int mid;

if(low < high) {

mid = (low + high) / 2;

sort(low, mid);

sort(mid+1, high);

merging(low, mid, high);

} else {

return;

}

}

int main() {

int i;

printf("List before sorting\n");

for(i = 0; i <= max; i++)

printf("%d ", a[i]);

sort(0, max);

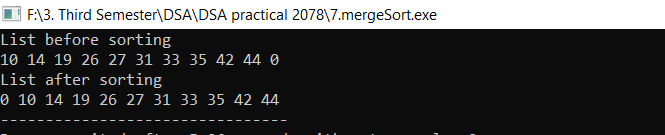
printf("\nList after sorting\n");

for(i = 0; i <= max; i++)

printf("%d ", a[i]);

return 0;

}



# **Write a C program to implement sequential search.**

#include <stdio.h>

int main()

{

int arr[]={12,23,78,98,67,56,45,19,65,9},key,i,flag=0;

printf("\nENTER A NUMBER: ");

scanf("%d",&key);

for(i=0;i<10;i++)

{

if(key==arr[i])

flag=1;

}

if(flag==1)

printf("\nTHE NUMBER %d EXISTS IN THE ARRAY",key);

else{

printf("\nTHE NUMBER %d DOES NOT EXIST IN THE ARRAY",key);

}

return 0;

}

