##### OBJECT ORIENTED PROGRAMMING LAB

##### LAB RECORD

###### ***Submitted by***

##### Name of the candidate

**Submitted to: Name of Faculty**



2020-2021

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY, AB ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA

**Table of Contents**

|  |  |
| --- | --- |
| **Lab Exercise with topic** | **Page No.** |
| Lab Exercise 1: Revisiting C | 3 |
| Lab Exercise 2: Revisiting C | 12 |
| Lab Exercise 3: Searching | 20 |
|  |  |
|  |  |
|  |  |
|  |  |

**Instructions:**

1. Write the text as follows:

(i) Font Type : Times New Roman

(ii) Font size : 12 points size

(iii) Spacing 1 in each line

(iv) Page No. Centre

2. Start new lab exercise from new page.

3. First write the problem or question then write the C/C++ program.

4. Follow proper indentation style while writing the program.

5. Add the watermark of your name and Er. No. on each page.

6. Write your own programs don’t copy-paste from any other source.

7. Lab record should have exactly same programs as you have submitted in classroom.

8. This is a sample Lab record.

**Lab Exercise 1: Revisiting C**

**Q1. WAP to create the linked list of n nodes.**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//This program is developed by XYZ (Er. No)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

#include <stdlib.h>

#include<conio.h>

struct Node

{

int INFO; //Data of the Node

struct Node \*NEXT; //Address of the next Node

};

struct Node\* reverse(struct Node\* start);

void selection\_sort(struct Node\*);

void bubble\_sort(struct Node\*);

struct Node \* createNodeList(int n); // function to create the list

void displayList(struct Node \*START); // function to display the list

void Delete\_From\_Beg(struct Node \*\*START);

int main()

{

int n;

printf(" Input the number of Nodes : ");

scanf("%d", &n);

struct Node \*START=createNodeList(n);

displayList(START);

//Delete\_From\_Beg(&START);

//START=reverse(START);

//printf("\nList after reverse: \n");

//displayList(START);

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

//selection\_sort(START);

bubble\_sort(START);

displayList(START);

getch();

return 0;

}

void Delete\_From\_Beg(struct Node \*\*START)

{

struct Node \*temp=\*START;

\*START=(\*START)->NEXT;

free(temp);

printf("\nList after deleting first node:\n");

displayList(\*START);

}

struct Node \*createNodeList(int n)

{

struct Node \* START=NULL;

struct Node \*New\_Node, \*temp;

int num, i;

if (n<=0) // for any value of n<=0 creat emmpty list

return NULL;

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

if(START == NULL) //check whether the fnNode is NULL and if so no memory allocation

{

printf(" Memory can not be allocated.");

}

else

{

// reads data for the Node through keyboard

printf(" Input data for Node 1 : ");

scanf("%d", &num);

START->INFO = num;

START->NEXT = NULL; // links the address field to NULL

temp = START;

// Creating n Nodes and adding to linked list

for(i=2; i<=n; i++)

{

New\_Node = (struct Node \*)malloc(sizeof(struct Node));

if(New\_Node == NULL)

{

printf(" Memory can not be allocated.");

break;

}

else

{

printf(" Input data for Node %d : ", i);

scanf(" %d", &num);

New\_Node->INFO = num; // put value in num field of New\_Node

New\_Node->NEXT = NULL; // links the address field of New\_Node with NULL

temp->NEXT = New\_Node; // links previous Node i.e. temp to the fnNode

temp = temp->NEXT;

}

}

}

return START;

}

void displayList(struct Node \*START)

{

struct Node \*temp;

if(START == NULL)

{

printf(" List is empty.");

}

else

{

printf("Linked list is: " );

temp = START;

while(temp != NULL)

{

printf("%d->", temp->INFO); // prints the data of current Node

temp = temp->NEXT; // advances the position of current Node

}

printf("NULL");

}

}

struct Node\* reverse(struct Node\* start)

{

struct Node\* p,\*c,\*n;

c=start;

n=start->NEXT;

start->NEXT=NULL;

while(n!=NULL)

{

p=c;

c=n;

n=n->NEXT;

c->NEXT=p;

}

return c;

}

void selection\_sort(struct Node \*start)

{

int min,t;

struct Node\* temp1,\*temp2,\*min\_temp;

for(temp1=start;temp1->NEXT!=NULL;temp1=temp1->NEXT)

{

min=temp1->INFO;

min\_temp=temp1;

for (temp2=temp1->NEXT;temp2!=NULL;temp2=temp2->NEXT)

{

if (min>temp2->INFO)

{

min=temp2->INFO;

min\_temp=temp2;

}

}

t=min\_temp->INFO;

min\_temp->INFO=temp1->INFO;

temp1->INFO=t;

}

}

void bubble\_sort(struct Node \*start)

{

int flag,t;

struct Node\* temp1,\*temp2,\*min\_temp;

for(temp1=start;temp1->NEXT!=NULL;temp1=temp1->NEXT)

{

flag=0;

for (temp2=start;temp2->NEXT!=NULL;temp2=temp2->NEXT)

{

if (temp2->INFO>temp2->NEXT->INFO)

{

t=temp2->INFO;

temp2->INFO=temp2->NEXT->INFO;

temp2->NEXT->INFO=t;

flag=1;

}

}

if (flag==0)

break;

}

}