LAB 6

Q1:

MAIN.C

```
#include <stdio.h>
#include<string.h>
#include <stdbool.h>
#include "file_opertn.h"
int main(){
  char *file="file.txt";
  if (checkfile(file)){
    printf("file opened");
  }
  writetext(file,"Hello World");
  readfile(file);
  return 0;
}
```

FILE_OPETN.H

```
#ifndef FILE_OPERATION_H
#define FILE_OPERATION_H
#include <stdbool.h>
bool checkfile(char *file);
void readfile(char *file);
void writetext(char *file, char *text);
#endif
```

FILE_OPERTN.C

```
#include "file_opertn.h"
#include <stdio.h>
```

```
#include <stdlib.h>
#include <stdbool.h>
bool checkfile(char *filename){
  FILE *file=fopen(filename,"r");
  if (file!=NULL){
    return true;
  }
  return false;
}
void readfile(char *filename) {
  FILE *file=fopen(filename,"r");
  char data[200];
  while (fgets(data,200,file)!=NULL){
    printf("%s",data);
  }
  fclose(file);
}
void writetext(char *filename, char *text){
  FILE *file=fopen(filename,"w");

  fprintf(file,"%s",text);
  fclose(file);
}
```

Q2:

MAIN.C

```c
#include "linkedlist.h"


int main() {
    struct Node *list = createLinkedList();
    list = insertAtBeginning(list, 5);
    list = insertAtBeginning(list, 10);
    displayLinkedList(list);
    struct Node *searchResult = searchElement(list, 20);
    freeLinkedList(list);
    return 0;
}
```

LINKEDLIST.H

```c
#ifndef LINKED_LIST_H
#define LINKED_LIST_H


struct Node {
    int data;
    struct Node *next;
};
struct Node *createLinkedList();
struct Node *insertAtBeginning(struct Node *head, int data);
struct Node *searchElement(struct Node *head, int data);
void displayLinkedList(struct Node *head);


void freeLinkedList(struct Node *head);
#endif
```

LINKEDLIST.C

```
#include "linkedlist.h"

#include <stdio.h>

#include <stdlib.h>




struct Node *createLinkedList() {

    return NULL;

}


 struct Node *insertAtBeginning(struct Node *head, int data) {

    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));

    if (newNode == NULL) {

        perror("Memory allocation error");

        exit(EXIT_FAILURE);

    }


    newNode->data = data;

    newNode->next = head;


    return newNode;

}

struct Node *searchElement(struct Node *head, int data) {

    struct Node *current = head;


    while (current != NULL) {

        if (current->data == data) {
            return current;
        }
        current = current->next;
    }

    return NULL;
}

void displayLinkedList(struct Node *head) {
    struct Node *current = head;

    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }

    printf("NULL\n");
}

void freeLinkedList(struct Node *head) {
    struct Node *current = head;
    struct Node *nextNode;

    while (current != NULL) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}
```