**First Semester 2023-24**

**Data Structures and Algorithms Design (Merged-SEZG519/SSZG519)**

**LAB 2 (Elementary Data Structures)**

*Notes: This lab covers the practicals on Elementary Data Structures, i.e., stack, queue, and linked list. We have mentioned programs in C language in the lab sheet. However, students are free to choose any of the programming languages to develop the solution to lab tasks.*

1. Complete the following code to develop Stack using Array. Do not forget to print a relevant message when the stack is full and the stack is empty.

|  |
| --- |
| #include <stdio.h>  #define MAXSIZE 5  int stack[MAXSIZE];//stack is an integer array for implementing Stack...  int tos = -1;//tos is index to top of the stack...  /\* Check if the stack is empty \*/  int isempty(){  *//Write your code here...*  }  /\* Function to return the topmost element in the stack \*/  int top(){  *//Write your code here...*  }  /\* Function to delete from the stack \*/  int pop(){  *//Write your code here...*  }  /\* Function to insert into the stack \*/  void push(int data){  *//Write your code here...*  }  /\* Main function \*/  void main(){  int data;  push(10);  data = pop();  printf("Deleted element %d\n",data);  data = pop();  push(21);  push(32);  data = pop();  printf("Deleted element %d\n",data);  push(43);  push(54);  printf("Element at top of the stack: %d\n" ,top());  printf("Deleting Elements: \n");  while(!isempty()) {  data = pop();  printf("%d\n",data);  }  } |

1. Complete the following code to develop Queue using Array. Do not forget to print a relevant message when the Queue is full and the Queue is empty.

|  |
| --- |
| #include <stdio.h>  #define MAXSIZE 5  int queue[MAXSIZE];  int front = -1;  int rear = -1;  /\* Check if the queue is empty \*/  int isEmpty(){  *//Write your code here...*  }  /\* Function to return the front element in the queue \*/  int front(){  *//Write your code here...*  }  /\* Function to insert into the queue \*/  void enqueue(int data){  *//Write your code here...*  }  /\* Function to delete from the queue \*/  int dequeue(){  *//Write your code here...*  }  void main(){  enqueue(10);  int data = dequeue();  printf("Deleted element %d\n",data);  data = dequeue();  printf("Deleted element %d\n",data);  enqueue(21);  enqueue(32);  data = dequeue();  printf("Deleted element %d\n",data);  enqueue(43);  enqueue(54);  enqueue(65);  enqueue(76);  while(!isEmpty()) {  data = dequeue();  printf("%d\n",data);  }  } |

1. Develop a code to implement a feature that highlights matching pairs of parentheses, brackets, and braces. A stack can be employed to track the opening and closing symbols and ensure proper nesting.

Sample input 1: {[()]}

Sample output 1: Parentheses are balanced.

Sample input 2: {[(}]}

Sample output 2: Parentheses are not balanced.

Sample input 3: {[()]

Sample output 3: Parentheses are not balanced.

|  |
| --- |
| #include <stdio.h>  #define MAXSIZE 5  char stack[MAXSIZE];//stack is an integer array for implementing Stack...  int tos = -1;//tos is index to top of the stack...  /\* Check if the stack is empty \*/  int isempty(){  *//Write your code here...*  }  /\* Function to delete from the stack \*/  char pop(){  if(isempty()){  printf("Stack is EMPTY\n");  return '0';  }  else{  *//Write your code here...*  }  }  /\* Function to insert into the stack \*/  void push(char data){  if(tos==MAXSIZE-1){  printf("Stack is FULL\n");  return;  }  *//Write your code here...*  }  // Check if an expression has balanced parentheses  int areParenthesesBalanced(const char \*expression) {  for (int i = 0; expression[i] != '\0'; i++) {  *//Write your code here...*  }  }  void main() {  const char \*expression = "{[()]}";  if (areParenthesesBalanced(expression)) {  printf("Parentheses are balanced.\n");  } else {  printf("Parentheses are not balanced.\n");  }  } |

1. Develop a code to output infix expression to prefix and postfix expressions. Your code should be able to show the error if the input infix expression is improper.

Sample Input 1: Infix: (A+B)^C-(D\*E)/F

Sample Output 1: Prefix: -^+ABC/\*DEF, Postfix: AB+C^DE\*F/-

Sample Input 2: Infix: (A+B^C-(D\*E)/F

Sample Output 2: Incorrect infix expression

1. Implement Tower of Hanoi Game using Stack with iterative method.
2. Code the functions of *enqueuerear*, *enqueuefront*, *dequeuefront*, *dequeuerear* for Double Ended Queue data structure.