# INDEX

Name Prabhanjan Bhat Sub.

Std.: 3rd Sem Div. Roll No. 196

Telephone No. E-mail ID.

Blood Group. Birth Day.

| Sr.No. | Title | Page No. | Sign./Remarks |
|---|---|---|---|
| 1. | 7/12/2023 (Practice Programs) | 10 | |
| 2. | 21/12/2023 (Stack Implementation) | | |
| 3. | 28/12/2023 (Infix to Postfix, Postfix Evaluation, Linear queue) | 10 | |
| 4. | 11/1/2024 (Circular queue, singly linked List) | 10 | 12/1/24 |
| 5. | 18/1/2024 Singly linked List (Deletion) Leetcode → (1) Leet cod → (2) | 10 | |
| 6. | 25/1/2024 Linked List operations, stack and queue implementation using linked List | 10 | |
| 7. | 1/2/2024 Doubly Linked List Leetcode → (3) | 10 | |
| 8. | 15/2/2024 Binary Search tree Leetcode → (4) | 10 | |
| 9. | 22/2/2024 BFS and DFS HackerRank Problem | 10 | 22/2/24 |
| 10. | 29/2/2024 Hashing program | 10 | 29/2/24 |

10

21/12/23 Lab-1

PAGE NO :
DATE :

1. Swapping 2 numbers

```
#include<stdio.h>
void swap(int *x, int *y);
void main()
{  int x, y;
   printf("enter the 2 nos");
   scanf("%d %d", &x, &y);
   printf("\n the numbers before swapping %d and %d",
                x, y);
   swap(&x, &y);
   return 0;
}
Void ~~swapptr~~ swap(int *x, int *y)
  { int temp = *x;
      *x = *y
      *y = temp;
      printf("\n the nos after swapping %d and
          %d", *x, *y);
}
```

Output: Enter the 2 nos: 20 10
The numbers before swapping 20 and 10
The nos after swapping 10 and 20

2. #WAP a prog to simulate the working of stack using an array with the following:
(a) Push
(b) Pop
(c) Display

```
#include<stdio.h>
#include<math.h>
#define SIZE 5

int stack[SIZE];
int top = -1;

void push(int element);
void pop();
void display();

int main() {
    int choice, element;
    
do {
        printf("\n Stack operations: \n");
        printf("1. Push \n");
        printf("2. Pop \n");
        printf("3. Display \n");
        printf("4. Exit \n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

switch(choice) {
  case 1: printf("Enter element to push: ");
          scanf("%d", &element);
          push(element);
          break;
```

```
case 2 : pop(); break;

case 3 : display(); break;

case 4 : printf("Exiting the program.\n"); break;

default : printf("Invalid choice. Please enter a valid
                 option.\n");
}
} while (choice != 4)
return 0;
}
void push (int element)
{   if (top == SIZE-1)
        { printf("Stack overflow. Cannot push.\n");
        }

    else
    { top = top+1;
      stack[top] = element;
    printf("%d pushed to stack.\n", element); }
}
void pop()
{     if(top == -1)
        { printf("Stack is empty. Nothing to
                 element display.\n"); }
    else {
        printf("%d popped from the stack.\n", stack[top]);
        top--;
    }
}
void display () {
    if (top == -1) {
        printf("Stack is empty. Nothing to display.\n");
}
```

underflow Cannot pop

```
else {
    printf("Stack elements: ");
    for(int i=0; i<=top ; i++)
    {
        printf("%d", stack[i]);
    }
    printf("\n");
}
}
```

Output: ~~Enter/ele~~

Stack operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1
Enter element to push: 56
56 pushed to the stack

~~Stack operati~~
Enter your choice: 1
Enter element to push: 34
34 pushed to the stack.

Enter your choice: 2
34 popped from the stack

Enter your choice: 3
Stack elements: 56

Enter your choice: 4
Exiting the program.

PAGE NO :
DATE :

3. WAP to implement dynamic memory allocation functions like malloc, free, calloc and realloc

```
#include <stdio.h>
int main()
{   int *ptr;
    int n;
    printf("Enter number of elements:");
    scanf("%d", &n);
    ptr = (int*) malloc(n * sizeof(int));

    if (ptr == NULL)
    {   printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated
                using malloc.\n");
        printf("Enter elements of array\n");

        for(int i=0; i<n; i++)
        {   scanf("%d", &ptr[i]);
    }
}
free(ptr);
 printf("Memory Freed\n");
 ptr = (int*) calloc(n, sizeof(int));

  if (ptr==NULL) {
        printf("Memory not allocated.\n");
         exit(0);
    }
else { printf("Memory successfully allocated
                using calloc.\n");
```

```
printf("Enter elements of array\n");
for(int i=0; i<n; i++)
{ scanf("%d", &ptr[i]);
}
printf("The elements of the array are:");
for(int i=0; i<n; i++)
{ printf("%d", ptr[i]);
}
int n1;
printf("Enter new size for realloc\n");
scanf("%d", &n);
ptr = (int *)realloc(ptr, n1 * sizeof(int));

printf("Memory successfully re-allocated
      using realloc.\n");
printf("Enter more variables:\n");

for(int i=n; i<n1; i++)
{ scanf("%d", &ptr[i]);
}

printf("The elements of the array are:");
for(int i=0; i<n1; i++)
{ printf("%d", ptr[i]);
}

free(ptr);

printf("Memory freed\n");
}
}
```

PAGE NO :
DATE :

Output :
Enter number of elements : 4
Memory successfully allocated using malloc.

Enter elements of array
12 23 45 67
The elements of the array are : 12, 23, 45, 67,
Memory Freed

Memory successfully ~~allocated~~ allocated using calloc.
Enter elements of array
12 23 45 65

The elements of the array are : 12, 23, 45, 65,
Enter new size for Realloc
5

Memory successfully re-allocated using realloc
Enter more variables :
6
The elements of the array are : 12, 23, 45, 65, 6,
Memory Freed.

21/12

Week-2 (Lab program 2)

PAGE NO :
DATE :

① Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators +, -, *, / & ^

```
#include<stdio.h>
#include<stdlib.h>

#define MAX_SIZE 100

int isOperator(char ch) {
  return (ch=='+' || ch=='-' || ch=='*' || ch=='/'
          || ch=='%');
}

int precedence(char operator) {
    if(operator=='+' || operator=='-')
        return 1;
    if(operator=='*' || operator=='/' || operator=='%')
        return 2;

    return 0;
}

void infixToPostfix(char infix[], char postfix[]) {
  char stack[MAX_SIZE];
  int top=-1;
  int i,j;

for(i=0, j=0; infix[i]!='\0'; i++) {
    if(infix[i]>='0' && infix[i]<='9')
  { postfix[j++]=infix[i];
  }
```

```
else if (isOperator(infix[i])) {
    while (top >= 0 && precedence(stack[top]) >= precedence(infix[i]))
    {
        postfix[j++] = stack[top--];
    }
    stack[++top] = infix[i];
}
else if (infix[i] == '(') {
    stack[++top] = infix[i]; }
else if (infix[i] == ')') {
    while (top >= 0 && stack[top] != '(') {
        postfix[j++] = stack[top--];
    }

    if(top >= 0 && stack[top] == '(') {
        top--;
    }
} // end of else if
} // end of for
while (top >= 0) {
    postfix[j++] = stack[top--];
}

postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];

    printf("Enter infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);
```

```
    printf("Postfix expression : %s\n", postfix);

    return 0;
}
```

Output: Enter infix expression : 2 + (3*4) - 5
Postfix expression : 2 3 4 * + 5 -

① Write a program to postfix evaluation

```
#include<stdio.h>
#include<stdlib.h>
#<ctype.h>

#define MAX_STACK_SIZE 100

int stack[MAX_STACK_SIZE];
int top = -1;

void push(int item) {
    if(top == MAX_STACK_SIZE - 1) {
        printf("Stack Overflow\n");
        exit(EXIT_FAILURE);
    }
    stack[++top] = item;
}

int pop() {
    if(top == -1) {
        printf("Stack Underflow\n");
        exit(EXIT_FAILURE);
    }
    return stack[top--];
}
```

PAGE NO :
DATE :

```
int isOperator(char ch) {
    return(ch=='+' || ch=='-' || ch=='*' || ch=='/' ||
           ch=='%');
}

int evaluatePostfix(char postfix[]){
    int i=0;
while (postfix[i] != '\0') {
char currentSymbol = postfix[i];

if(isdigit(currentSymbol)){
    push(currentSymbol - '0');
}
else if(isOperator(currentSymbol)) {
 int operand2 = pop();
 int operand1 = pop();

switch(currentSymbol) {

case '+': push(operand1 + operand2);
          break;

case '-': push(operand1 - operand2);
          break;

case '*': push(operand1 * operand2);
          break;

case '/': push(operand1 % operand2);
          break;
 }
}
i++; //for while loop
}
return pop();
}
```

```
int main()
{   char postfixExpression[100];

    printf("Enter postfix expression: ");
    scanf("%s", postfixExpression);

    int result = evaluatePostfix(postfixExpression);

    printf("Result: %d\n", result);

    return 0;
}
```

Output: Enter postfix expression: 234*+5-
Result: 9

3(a) Queue Implementation

```
#include<stdio.h>
#include<stdlib.h>
#define max 20

int queue[max];
int front=-1, rear=-1;

void enqueue(int a);
int dequeue();
void display();
```

```
void main() {
    int n, m;
    printf("Enter \n 1. to insert \n 2. to remove
        \n 3. to display \n");
    scanf("%d", &n);
    while(n!=3) {
        switch(n) {
        case 1: printf("Enter element:");
                scanf("%d", &m);
                enqueue(m); break;
        case 2: m = dequeue();
                printf("Element removed is %d", m);
                break;
        case 3: display();
        default: printf("Invalid Input");
    }
}
}

void enqueue(int a) {
    if(rear == max-1) {
        printf("Queue full");
    }
    rear = -1;
    queue[rear] = a;
}

int dequeue() {
    if(rear == -1 || front == rear) {
        printf("Queue empty");
    }
    else { front += 1;
        return queue[front];
    }
}
```

PAGE NO :
DATE :

```
void display() {
 printf("Queue :");
for(int i=front+1; i<=rear; i++) {
    printf("%d\t", queue[i]);
  }
}
```

Output: 1. Insert element
2. Delete element
3. Exit
1
Enter element : 23

1. Insert Element
2. Delete element
3. Exit
1
Enter element : 54

1. Insert Element
2. Delete element
3. Exit
2
Deleted element : 23

1. Insert Element
2. Delete element
3. Exit
3

# Week-3 Lab 3

PAGE NO :
DATE :

3(b) Circular Queue

```
#include<stdio.h>
#define MAX 6
int Q[MAX];
int front = -1, rear = -1;

void enqueue(int element);
void dequeue();
void display();

int main(){ int choice, element;

while(1){ printf("Enter 1 to insert, 2 to delete,
3 to display, 4 to exit : ");

switch(choice){
  case 1: printf("Enter the element to insert:");
          scanf("%d", &element);
          enqueue(element); break;

case 2: dequeue(); break;

case 3: display(); break;

case 4: printf("Exiting the program.\n");
        return 0;

default: printf("Invalid choice. Please enter a
         valid option.\n");
}
}
}
```

```
void enqueue (int element) {
  if ((rear+1) % MAX == front) {
    printf ("Queue overflow. Cannot insert element.\n");
    return;
  }
  else if (front == -1) { front = 0;
                          rear = (rear+1) % MAX;
  Q[rear] = element;
  printf("Element %d inserted successfully.\n", element);
  }
}
void dequeue() {
    if (front == -1) {
      printf("Queue underflow. Cannot delete element.\n");
    }
    else {
      printf("Deleted element: %d\n", Q[front]);
      if (front = rear) {
          front = rear-1;
      } else {
          front = (front+1) % MAX;
      }
}
void display() {
    if (front == -1) {
      printf("Queue is empty.\n");
      return;
    }
    else {
      for (int i = front; i != (rear+1) % MAX; i = (i+1) % MAX)
          printf("%d\n", Q[i]);
    }
```

PAGE NO :
DATE :

Output : Enter 1 to insert, 2 to delete, 3 to display,
4 to exit : 1
Enter the element to insert : 20
Element 20 inserted successfully.
Enter 1 to insert, 2 to delete, 3 to display,
4 to exit : 1
Enter the element to insert : 40
Element 40 inserted successfully.
Enter 1 to insert, 2 to delete, 3 to display,
4 to exit : 2
Deleted element : 20
Enter 1 to insert, 2 to delete, 3 to display,
4 to exit : 3
40
Enter 1 to insert, 2 to delete, 3 to display,
4 to exit : 4
Exiting the program

4. Singly linked List implementation:

```
#include<stdio.h>
#include<stdlib.h>
void create();
void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();

struct node *head = NULL;

struct node { int data;
              struct node *next; };
int main()
{ int choice;
  while (1) { printf("0. Create \n");
              printf("1. display \n");
              printf("2. Insert Node at beginning\n");
              printf("3. Insert Node in specific pos\n");
    printf("4. Insert Node at end of LinkedList \n");
    printf("5. Delete Node at beginning \n");
    printf("6. Delete Node at end \n");
    printf("7. Delete Node at position \n");
    printf("8. **To exit** ");

    printf("\n Enter your choice: ");
    scanf("%d", &choice);

    switch(choice)
```

```
{ case 0 : create(); break;
  case 1 : display(); break;
  case 2 : insert_begin(); break;
  case 3 : insert_pos(); break;
  case 4 : insert_end(); break;
  case 5 : delete_begin(); break;
  case 6 : delete_end(); break;
  case 7 : delete_pos(); break;
  case 8 : exit(0); return 0; break;
  default : printf("\n Wrong choice"); break;
}
}}
void create() {
  struct node *temp;
  temp=(struct node*) malloc(sizeof(struct node));
  printf("Enter node data:");
  scanf("%d", &temp->data);
  temp->next=NULL;

  if(head==NULL) {
      head=temp; }
  else {
      struct node *ptr=head;
      while(ptr->next=NULL)
      { ptr=ptr->next; }
      ptr->next=temp;
}}
void display()
{ if(head==NULL)
  { printf("Linked List is Empty\n");
    return;
  }
  printf("Linked list:");
```

```
struct node *ptr = head;
while(ptr != NULL)
{
    printf("%d", ptr->data);
    ptr = ptr->next;
}
printf("\n"); }

void insert_begin() {
struct node *temp;
temp = (struct node *) malloc (sizeof(struct node));
printf("Enter node date:");
scanf("%d", &temp->data);
temp->next = NULL;

if (head == NULL)
{   head = temp;
    return;
}
else
{ temp->next = head;
    head = temp;
  }
}

void insert_pos() {
    struct node *temp;
    temp = (struct node*) malloc (sizeof(struct node));
printf("Enter node data:");
scanf("%d", &temp->data);
temp->next = NULL;

if (head == NULL) {
    head = temp;
    return;
```

```
else { struct node **prev_ptr;
       struct node *ptr = head;
       int pos;
       printf("Enter position: ");
       scanf("%d", &pos);
       for(int i=0; i<pos; i++)
       {
           prev_ptr = ptr;
           ptr = ptr->next; }
       temp -> next = ptr;
       prev_ptr -> next = temp;
   }
}
void insert_end() {
       struct node *temp;
       temp = (struct node *) malloc (sizeof(struct node));
   printf("Enter node data: ");
   scanf("%d", &temp->data);
   temp -> next = NULL;

    if(head==NULL)
   {   head = temp;
       return; }
   else {  struct node *ptr = head;
           while(ptr->next!=NULL)
           {  ptr = ptr->next; }
           ptr->next = temp;
        }
}
void delete_begin()
{  if(head==NULL)
    {  printf("Linked List is empty! Nothing to delete\n");
       return;
    }
```

```
else { struct node *ptr = head;
       head = head->next;
       free(ptr);
       printf("Node Deleted \n");
     }
}
void delete_end()
{  if(head == NULL)
   {  printf("Linked List is empty | Nothing to del
      return;
   }
   else if(head->next == NULL)
{  struct node *ptr = head;
      head = ptr->next;
      free(ptr);
}
   else { struct node *ptr = head;
          struct node *prevptr = NULL;
      while(ptr->next) = NULL)
      {  prev_ptr = ptr;
         ptr = ptr->next;
      }
   prev_ptr->next = NULL;
    free(ptr);
   }
}
void delete_pos()
  {  int pos;
     printf("Enter node position to delete:");
     scanf("%d", &pos);
     struct node *ptr = head;
     if(head == NULL)
     {  printf("Linked List is empty\n");
        return;
     }
```

```
else if(pos==0)
  {   ptr = head;
      head = ptr -> next;
      free(ptr);
  }
else {   struct node *prev_ptr;
         for(int i=0; i<pos; i++)
           {     prev_ptr = ptr;
                 ptr = ptr -> next;
           }
     prev_ptr -> next = ptr -> next;
        free(ptr);
  }
}
```

Output:

0. Create
1. Display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position

Enter your choice: 0
Enter node data: 60

0. Create
"

Enter your choice: 2
Enter node data: 70

0. Create
"

Enter your choice: 3
Enter node data: 80

Enter position: 1

0. Create
" " "
Enter your choice: 4
Enter node data: 30

0. Create
" " "
Enter your choice: 5
Node Deleted

0. Create
" " "
Enter your choice: 1
LinkedList: 80 60 30

0. Create
" " "
Enter your choice: 8

12/1/24