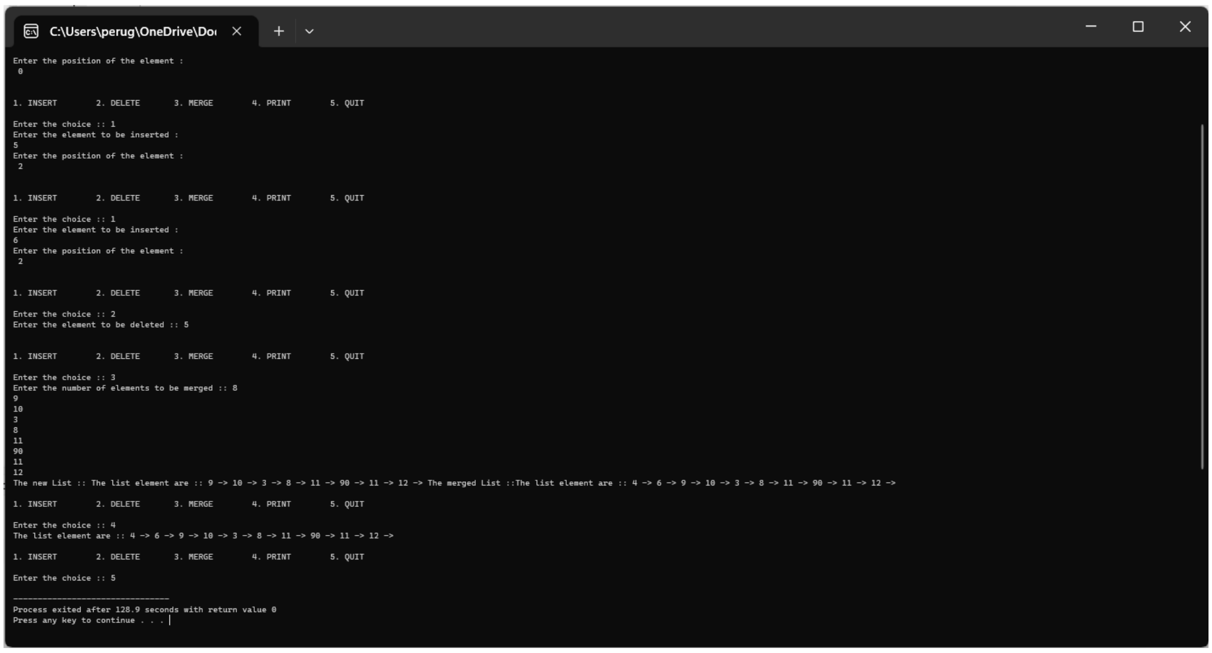
Data structure day-3:

1.write c program in single list following operation?

```
#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 :\n");
         scanf("%d",&x);
         printf("Enter the position of the element :\n ");
         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;
}
```

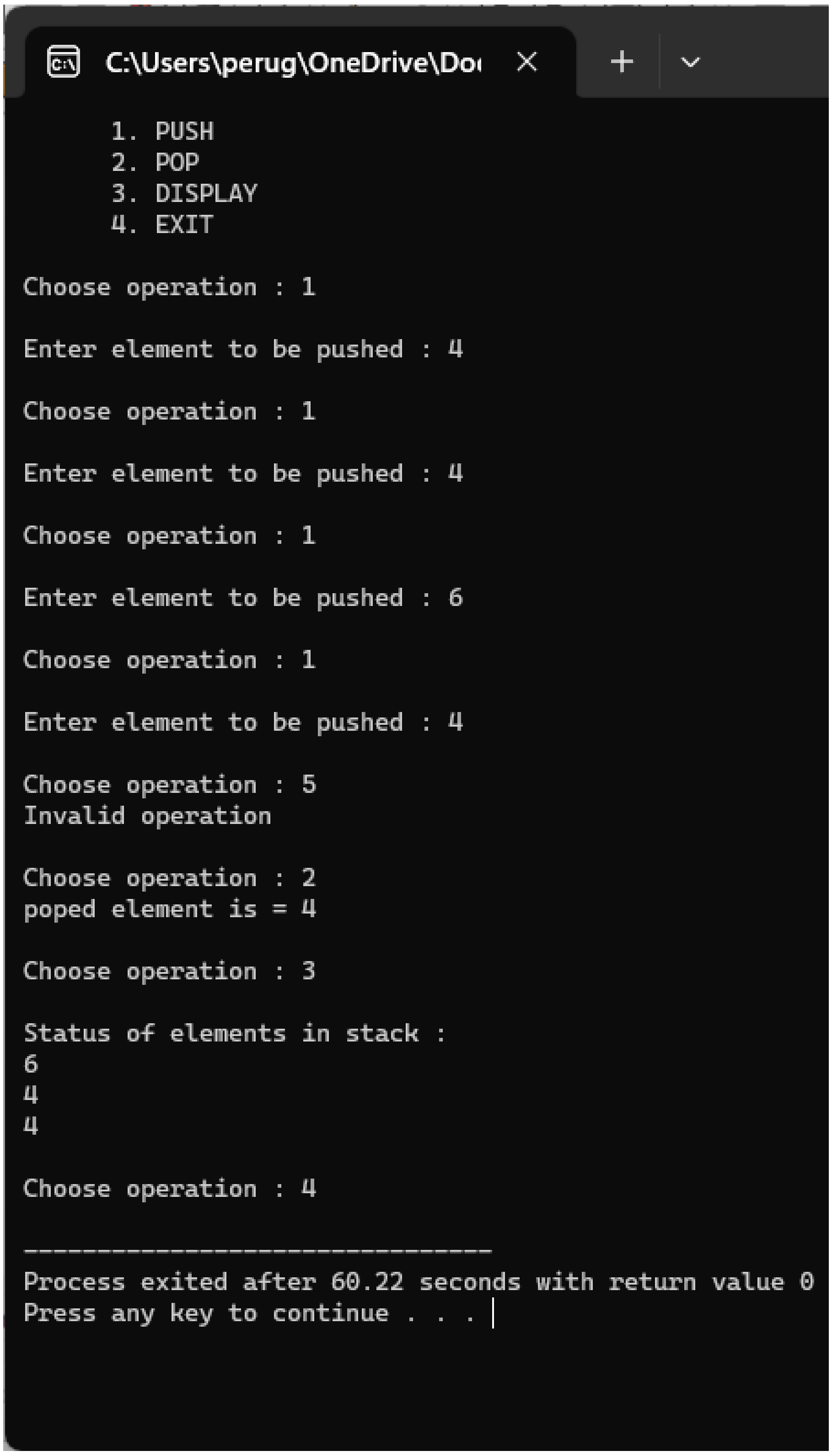


```
Enter the position of the element :
 0


1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 1
Enter the element to be inserted :
5
Enter the position of the element :
 2


1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 1
Enter the element to be inserted :
6
Enter the position of the element :
 2


1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 2
Enter the element to be deleted :: 5


1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 3
Enter the number of elements to be merged :: 8
9
10
3
8
11
90
11
12
The new List :: The list element are :: 9 -> 10 -> 3 -> 8 -> 11 -> 90 -> 11 -> 12 -> The merged List ::The list element are :: 4 -> 6 -> 9 -> 10 -> 3 -> 8 -> 11 -> 90 -> 11 -> 12 ->

1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 4
The list element are :: 4 -> 6 -> 9 -> 10 -> 3 -> 8 -> 11 -> 90 -> 11 -> 12 ->

1. INSERT        2. DELETE       3. MERGE        4. PRINT        5. QUIT

Enter the choice :: 5

--------------------------------
Process exited after 128.9 seconds with return value 0
Press any key to continue . . .
```

2.**write c program in stack following operation?**

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 5
struct stack
{
    int stk[MAXSIZE];
    int top;
};
typedef struct stack ST;
ST s;
void push ()
{
    int num;
    if (s.top == (MAXSIZE - 1))
    {
        printf ("Stack is Full\n");
        return;
    }
    else
    {
        printf ("\nEnter element to be pushed : ");
        scanf ("%d", &num);
        s.top = s.top + 1;
        s.stk[s.top] = num;
    }
    return;
}
int pop ()
{
    int num;
    if (s.top == - 1)
    {
        printf ("Stack is Empty\n");
        return (s.top);
    }
    else
    {
        num = s.stk[s.top];
        printf ("poped element is = %d\n", s.stk[s.top]);
        s.top = s.top - 1;
    }
    return(num);
}
void display ()
{
    int i;
    if (s.top == -1)
    {
        printf ("Stack is empty\n");
        return;
    }
    else
    {
        printf ("\nStatus of elements in stack : \n");
        for (i = s.top; i >= 0; i--)
        {
            printf ("%d\n", s.stk[i]);
        }
    }
}
int main ()
{
    int ch;
    s.top = -1;     printf ("\tSTACK OPERATIONS\n");
    printf("----------------------------\n");
    printf("     1. PUSH\n");
    printf("     2. POP\n");
    printf("     3. DISPLAY\n");
    printf("     4. EXIT\n");
    //printf("----------------------------\n");
    while(1)
    {
        printf("\nChoose operation : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                push();
            break;
            case 2:
                pop();
            break;
            case 3:
                display();
            break;
            case 4:
                exit(0);
            default:
                printf("Invalid operation \n");
        }
    }
```

```
    return 0;
}
```



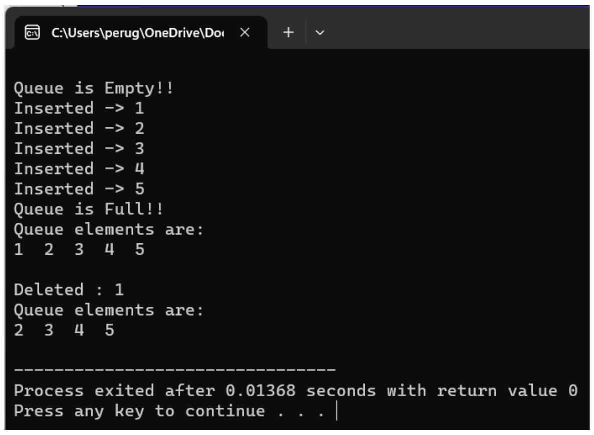
C:\Users\perug\OneDrive\Do

```
       1. PUSH
       2. POP
       3. DISPLAY
       4. EXIT

Choose operation : 1

Enter element to be pushed : 4

Choose operation : 1

Enter element to be pushed : 4

Choose operation : 1

Enter element to be pushed : 6

Choose operation : 1

Enter element to be pushed : 4

Choose operation : 5
Invalid operation

Choose operation : 2
poped element is = 4

Choose operation : 3

Status of elements in stack :
6
4
4

Choose operation : 4

--------------------------------
Process exited after 60.22 seconds with return value 0
Press any key to continue . . .
```

3.**write c program to implement queue following operation?**

```
#include <stdio.h>
#define SIZE 5
void enQueue(int);
void deQueue();
void display();

int items[SIZE], front = -1, rear = -1;

int main() {
  deQueue();
  enQueue(1);
  enQueue(2);
  enQueue(3);
  enQueue(4);
  enQueue(5);
  enQueue(6);

  display();
  deQueue();
  display();

  return 0;
}
```

```
void enQueue(int value) {
  if (rear == SIZE - 1)
    printf("\nQueue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = value;
    printf("\nInserted -> %d", value);
  }
}

void deQueue() {
  if (front == -1)
    printf("\nQueue is Empty!!");
  else {
    printf("\nDeleted : %d", items[front]);
    front++;
    if (front > rear)
      front = rear = -1;
  }
}
void display() {
  if (rear == -1)
    printf("\nQueue is Empty!!!");
  else {
    int i;
    printf("\nQueue elements are:\n");
    for (i = front; i <= rear; i++)
      printf("%d  ", items[i]);
  }
  printf("\n");
}
```



```
Queue is Empty!!
Inserted -> 1
Inserted -> 2
Inserted -> 3
Inserted -> 4
Inserted -> 5
Queue is Full!!
Queue elements are:
1  2  3  4  5

Deleted : 1
Queue elements are:
2  3  4  5

--------------------------------
Process exited after 0.01368 seconds with return value 0
Press any key to continue . . .
```

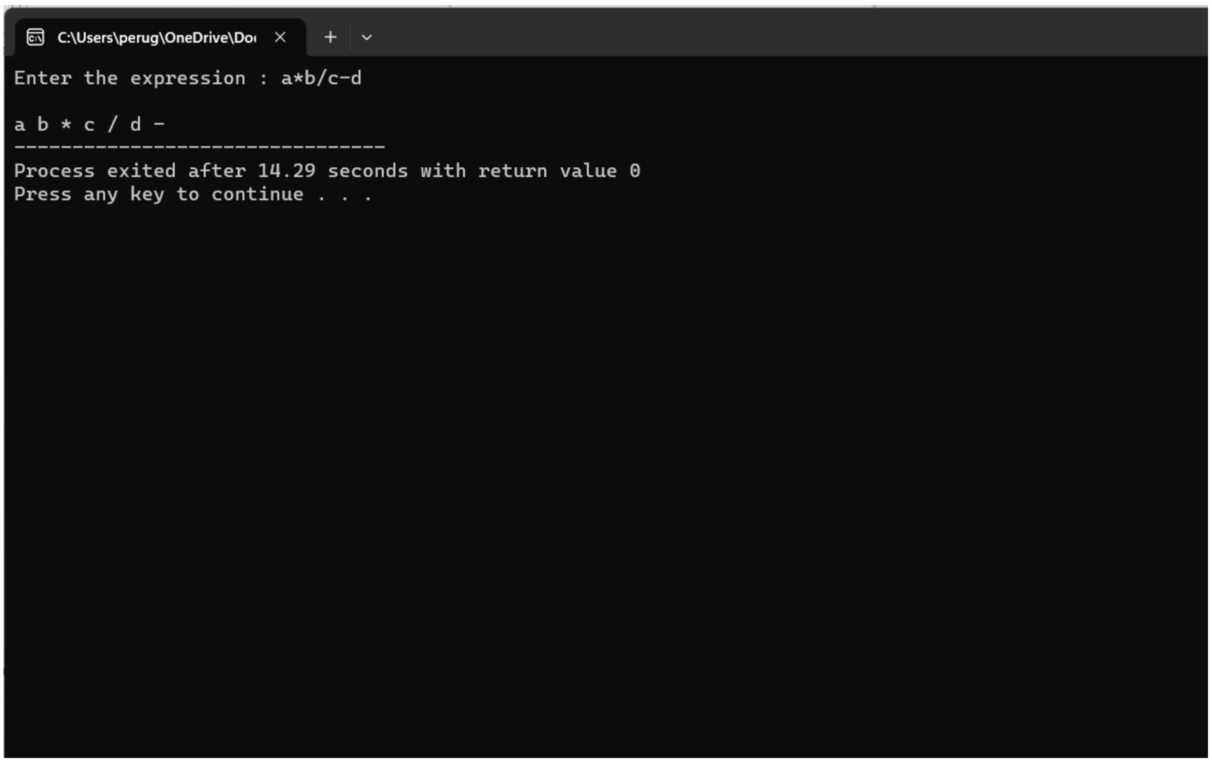
4. Write c program to convert infix expression to post fix expression using stack?

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

char stack[100];
int top = -1;

void push(char x)
{
    stack[++top] = x;
}
```

```
char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
```

```
printf("\n");
e = exp;

while(*e != '\0')
{
   if(isalnum(*e))
      printf("%c ",*e);
   else if(*e == '(')
      push(*e);
   else if(*e == ')')
   {
      while((x = pop()) != '(')
         printf("%c ", x);
   }
   else
   {
      while(priority(stack[top]) >= priority(*e))
         printf("%c ",pop());
      push(*e);
   }
   e++;
}

while(top != -1)
{
   printf("%c ",pop());
```

```
    }return 0;
}
```

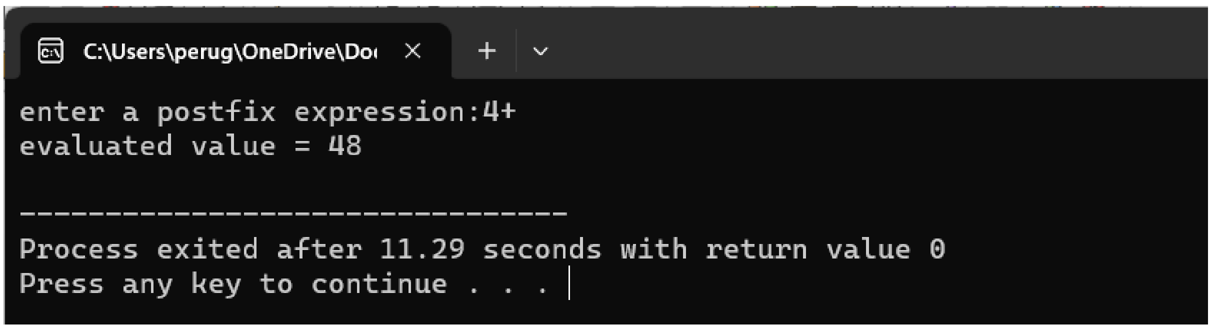


```
C:\Users\perug\OneDrive\Do
Enter the expression : a*b/c-d

a b * c / d -
--------------------------------
Process exited after 14.29 seconds with return value 0
Press any key to continue . . .
```

5.Write c program to evalute given expression by using stack?

```
#include<stdio.h>
int top = -1, stack [100];
main ( ){
  char a[50], ch;
  int i,op1,op2,res,x;
  void push (int);
  int pop( );
  int eval (char, int, int);
  printf("enter a postfix expression:");
  gets (a);
  for(i=0; a[i]!='\0'; i++){
    ch = a[i];
```

```
        if (ch>='0' && ch<='9')
            push('0');
        else{
            op2 = pop ( );
            op1 = pop ( );
            res = eval (ch, op1, op2);
            push (res);
        }
    }
    x = pop ( );
    printf("evaluated value = %d", x);
    gets(a);
}
void push (int n){
    top++;
    stack [top] = n;
}
int pop ( ){
    int res ;
    res = stack [top];
    top--;
    return res;
}
int eval (char ch, int op1, int op2){
    switch (ch){
        case '+' : return (op1+op2);
        case '-' : return (op1-op2);
        case '*' : return (op1*op2);
        case '/' : return (op1/op2);
    }
}
```



C:\Users\perug\OneDrive\Do

```
enter a postfix expression:4+
evaluated value = 48

--------------------------------
Process exited after 11.29 seconds with return value 0
Press any key to continue . . .
```