Integer Stack

#include<stdio.h>

int top = -1;

int stacks[3];

void push(int element){

    if (top == 2)

    {

        printf("PUSH: Stack is full\n");

    }

    else{

        top++;

        stacks[top] = element;

        printf("%d is inserted in stack\n",element);

    }

}

void peek(){

    if (top == -1)

    {

        printf("PEEK: Stack is empty.\n");

    }

    else{

    printf("The element %d is at the top of the stack\n",stacks[top]);

    }

}

void pop(){

    if(top == -1){

        printf("POP: Stack is empty \n");

    }

    else{

        printf("The element %d is removed from the stack\n",stacks[top]);

        top--;

    }

}

int main(int argc, char const \*argv[])

{

    push(10);

    push(20);

    push(30);

    push(40);

    peek();

    pop();

    pop();

    pop();

    pop();

    peek();

    return 0;

}

String stack

#include <stdio.h>

int top = -1;

char stacks [3];

void push (char number){

if (top==2){

    printf("%c cannot be inserted into a stack as stack is full.\n",number);

}

else{

top++;

stacks [top] = number;

printf ("%c is inserted into stack.\n",number);

}

}

void peek(){

    if (top == -1)

    {

        printf("PEEK: Stack is empty.\n");

    }

    else{

    printf("The element %d is at the top of the stack\n",stacks[top]);

    }

}

char pop (){

    if (top==-1){

        printf("Stack is empty.\n");

    }

    else{

        char r\_number=stacks[top];

    top--;

    printf ("%c",r\_number);

    }

}

int main(int argc, char const \*argv[])

{

push('1asd');

push('2asd');

push('3asd');

push('4asd');

pop();

pop();

pop();

    return 0;

}

Integer queue

#include<stdio.h>

int queues[3];

int rear = -1;

int front = -1;

void enqueue(int element){

    if (rear==2){

        printf("%d cannot be inserted into the queue. The queue is full.\n",element);

    }else{

        rear++;

        queues[rear] = element;

        printf("%d is in the queue\n",queues[rear]);

    }

}

void dequeue(){

    if(front == rear){

        printf("cant be dequed");

    }

    else{

        front++;

        printf("%d is dequeued \n",queues[front]);

    }

}

int main(int argc, char const \*argv[])

{

    enqueue(10);

    enqueue(20);

    enqueue(30);

    enqueue(40);

    dequeue();

    dequeue();

    dequeue();

    dequeue();

    return 0;

}

String queue

#include<stdio.h>

char \*queues[3];

int rear = -1;

int front = -1;

void enqueue(char element[]){

    if (rear==2){

        printf("%s cannot be inserted into the queue. The queue is full.\n",element);

    }else{

        rear++;

        queues[rear] = element;

        printf("%s is in the queue\n",queues[rear]);

    }

}

void dequeue(){

    if(front == rear){

        printf("cant be dequed");

    }

    else{

        front++;

        printf("%s is dequeued \n",queues[front]);

    }

}

int main(int argc, char const \*argv[])

{

    enqueue("1asd");

    enqueue("2asd");

    enqueue("3asd");

    enqueue("4asd");

    dequeue();

    dequeue();

    dequeue();

    dequeue();

    return 0;

}

Linkedlist

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

typedef struct Node{

    char data[50];

    struct Node \*nextnode;

}node;

node \*firstnode;

node \*newNode;

//insert from the beginning

void nodeFromTheBegining(char data[]){

    if (firstnode==NULL){

        firstnode=malloc(sizeof(node));

        strcpy(firstnode->data,data);

        printf("%s \n",data);

        firstnode->nextnode=NULL;

    }else{

        node \*newnode= malloc(sizeof(node));

        strcpy(newnode->data,data);

        printf("%s \n",data);

        newnode->nextnode=firstnode;

        firstnode=newnode;

    }

}

// insertion at the End

void insertANodeInTheEnd(char data[]){

    if (firstnode == NULL)

    {

        firstnode = malloc(sizeof(node));

        strcpy(firstnode->data,data);

        // printf("%s\n",data);

    }

    else{

    //traverse in the last node first

    node \*temp = firstnode;

    while (temp->nextnode != NULL)

    {

        temp = temp -> nextnode;

    }

    //else part

        newNode= malloc(sizeof(node));

        strcpy(newNode->data,data);

        temp->nextnode = newNode;

    }

}

//transverse from the end

void traverse(){

    node \*temp= firstnode;

    while (temp->nextnode!=NULL)

    {

        printf("%s ->",temp->data);

        temp =temp->nextnode;

    }

    printf("%s\n",temp->data);

}

//deletion from the end

void deleteANodeFromTheEnd(){

    node \*temp = firstnode;

    node \*secondLastNode;

    while (temp->nextnode != NULL)

    {

        // printf("%s ->",temp->data);

        secondLastNode = temp;

        temp = temp -> nextnode;

    }

    secondLastNode->nextnode = NULL;

}

void deleteANodeFromTheStart(){

    if(firstnode == NULL){

        printf("LinkedList is empty\n");

    }

    else{

        node \*secondNode = firstnode->nextnode;

        firstnode=secondNode;

    }

}

int main(int argc, char const \*argv[])

{

    // printf("IIMS College - DSA youtube channel playlist\n");

    // printf("Insertion operation.....\n");

    nodeFromTheBegining("Week 1 ");

    nodeFromTheBegining("Week 2");

    nodeFromTheBegining("Week 3");

    nodeFromTheBegining("Week 4");

    nodeFromTheBegining("Week 5");

    // traverse

    // printf("Traversing operation......\n");

    traverse();

//    printf("Deletion operation(Sam finished two of the to-do list)\n")

    deleteANodeFromTheStart();

    traverse();

    insertANodeInTheEnd("Week 5");

    traverse();

    deleteANodeFromTheEnd();

    traverse();

    return 0;

}