**R.Harini**

**18BCE1010**

a)Implementing stacks using arrays

#include<stdio.h>

#define MAX 5

int stack[20],top=-1;

int is\_empty(int\* s){

if (top==-1){

return 1;

}

else{

return 0;

}

}

int is\_full(int\* s){

if (top==MAX){

return 1;

}

else{

return 0;

}

}

void push(int\* s, int i){

if (is\_full(s)){

printf("Stack is full\n");

}

else{

top++;

s[top]=i;

}

}

void pop(int\*s){

if (is\_empty(s)){

printf("Stack is empty\n");

}

else{

printf("%d\n",s[top]);

top--;

}

}

void display(int\* s){

if (is\_empty(s)){

printf("Stack is empty\n");

}

else{

printf("Displaying the stack:\n");

for (int i=0;i<top+1;i++){

printf("%d\n",s[i]);

}

}

}

int main(){

push(stack,5);

push(stack,4);

push(stack,3);

push(stack,2);

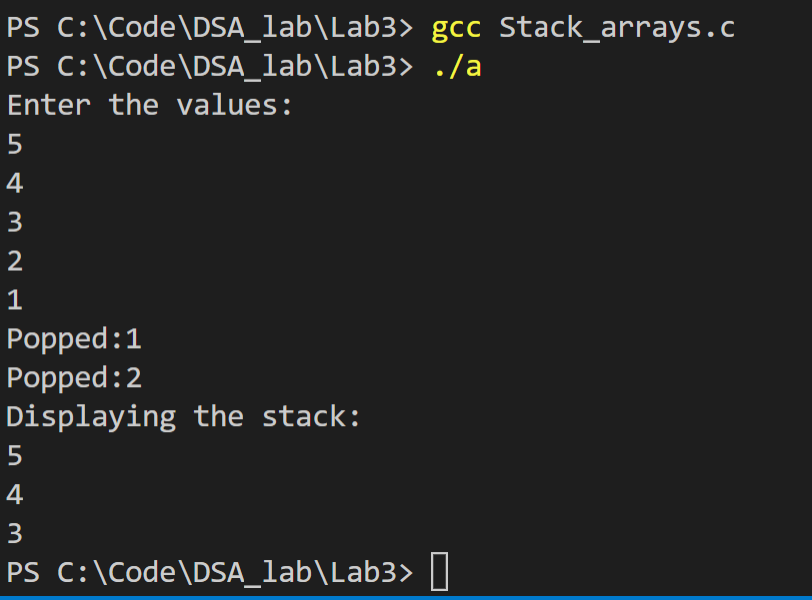
push(stack,1);

pop(stack);

pop(stack);

display(stack);

}



b) Balancing Parenthesis

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAX 5

struct Stack{

int top;

char a[30];

};

int is\_empty(struct Stack\* s){

if (s->top==-1){

return 1;

}

else{

return 0;

}

}

int is\_full(struct Stack\* s){

if (s->top==MAX){

return 1;

}

else{

return 0;

}

}

void push(struct Stack\* s, char i){

if (is\_full(s)){

printf("Stack is full\n");

}

else{

s->top++;

s->a[s->top]=i;

}

}

void pop(struct Stack\* s){

if (is\_empty(s)){

printf("Stack is empty\n");

}

else{

//printf("%c",s->a[s->top]);

s->top--;

}

}

void display(struct Stack\* s){

for (int i=0;i<(s->top+1);i++){

printf("%c\n",s->a[i]);

}

}

int main(){

struct Stack\* s=(struct Stack\*)malloc(sizeof(struct Stack));

s->top=-1;

char p[30];

scanf("%s",p);

for (int i=0;i<strlen(p);i++){

if (p[i]=='('){

push(s,p[i]);

}

else if (p[i]==')'){

pop(s);

}

else{

continue;

}

}

if (is\_empty(s)){

printf("Balanced");

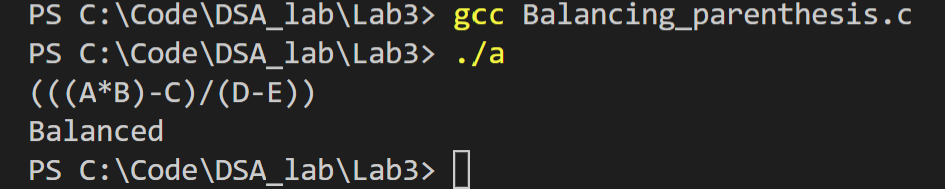
}

else{

printf("Not balanced");

}

}



c) Implementing Queues using arrays

#include<stdio.h>

#define MAX 5

int queue[30],front=-1, rear=-1;

int is\_empty(int\* q){

if (front==rear && front==-1 ){

return 1;

}

else{

return 0;

}

}

int is\_full(int\* q){

if (rear==MAX-1){

return 1;

}

else{

return 0;

}

}

void enqueue(int\* q, int i){

if (is\_full(q)){

printf("Queue is full\n");

}

else{

rear++;

q[rear]=i;

}

}

void dequeue(int\* q){

if (is\_empty(q)){

printf("Queue is empty\n");

}

else{

front++;

printf("%d\n",q[front]);

}

}

void display(int\* q){

if (is\_empty(q)){

printf("Queue is empty\n");

}

else{

printf("Displaying the queue:\n");

for (int i=front+1;i<rear+1;i++){

printf("%d\n",q[i]);

}

}

}

int main(){

enqueue(queue,5);

enqueue(queue,4);

enqueue(queue,3);

enqueue(queue,2);

enqueue(queue,1);

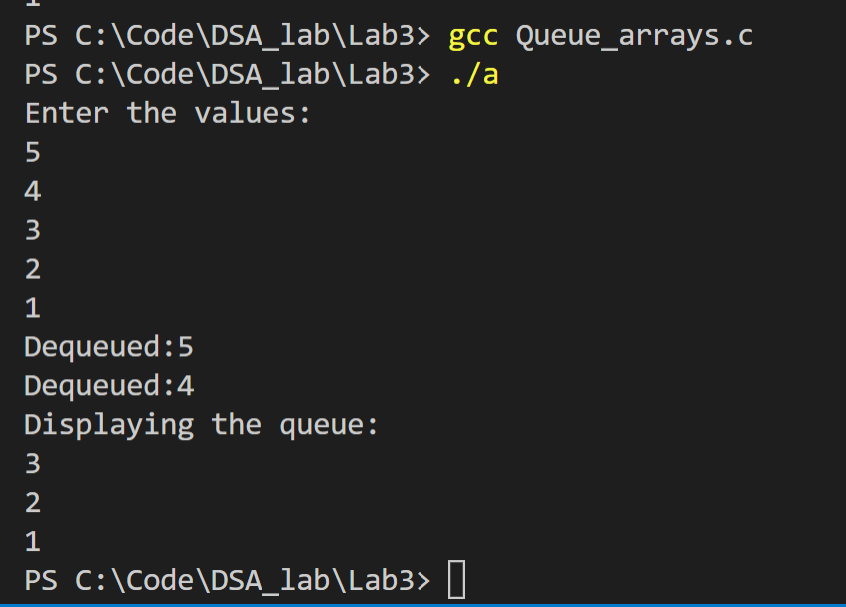
dequeue(queue);

dequeue(queue);

display(queue);

return 0;

}



d) Word problem using Disks (uses stacks and queues)

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX 5

struct Disks{

char type[10];

int ID;

int year;

char mname[10];

};

struct Stack{

int top;

struct Disks\* a[100];

};

int is\_empty(struct Stack\* s){

if (s->top==-1){

return 1;

}

else{

return 0;

}

}

int is\_full(struct Stack\* s){

if (s->top==(MAX-1)){

return 1;

}

else{

return 0;

}

}

void push(struct Stack\* s, struct Disks\* d){

if (is\_full(s)){

printf("Stack is full\n");

}

else{

s->top++;

s->a[s->top]=d;

}

}

struct Disks\* pop(struct Stack\* s){

if (is\_empty(s)){

printf("Stack is empty\n");

}

else{

struct Disks\* d=(struct Disks\*)malloc(sizeof(struct Disks));

d=s->a[s->top];

//printf("%s\n%d\n%d\n%s\n", d->type, d->ID, d->year, d->mname);

s->top--;

return d;

}

}

void display\_s(struct Stack\* s){

for (int i=0;i<(s->top+1);i++){

struct Disks\* d=(struct Disks\*)malloc(sizeof(struct Disks));

d=s->a[i];

printf("%s\n%d\n%d\n%s\n ", d->type, d->ID, d->year, d->mname);

}

}

struct Queue{

int front;

int rear;

struct Disks\* a[50];

};

int qis\_full(struct Queue\* q){

if (q->front==q->rear && q->rear>-1){

return 1;

}

else{

return 0;

}

}

int qis\_empty(struct Queue\* q){

if (q->rear==-1){

return 1;

}

else{

return 0;

}

}

void enqueue(struct Queue\* q, struct Disks\* d){

if (qis\_full(q)){

printf("Queue is full\n");

}

else{

q->rear++;

q->a[q->rear]=d;

}

}

void dequeue(struct Queue\* q){

if (qis\_empty(q)){

printf("Queue is empty\n");

}

else{

q->front++;

struct Disks\* d=(struct Disks\*)malloc(sizeof(struct Disks));

d=q->a[q->front];

printf("%s\n%d\n%d\n%s\n ", d->type, d->ID, d->year, d->mname);

}

}

void display\_q(struct Queue\* q){

if (qis\_empty(q)){

printf("Queue is empty\n");

}

else{

for (int i=(q->front+1);i<(q->rear+1);i++){

struct Disks\* d=(struct Disks\*)malloc(sizeof(struct Disks));

d=q->a[i];

printf("%s\n%d\n%d\n%s\n ", d->type, d->ID, d->year, d->mname);

}

}

}

void check\_year(struct Queue\* q, struct Disks\* d){

if (d->year==2015 && strcmp(d->type, "CD")==0){

printf("\n%s is from 2015\n",d->mname);

enqueue(q,d);

}

}

void sort\_disks(struct Disks\* d,int n){

struct Disks t;

for (int i=0;i<n;i++){

for (int j=i+1;j<n;j++){

if (d[i].year>d[j].year){

t=d[i];

d[i]=d[j];

d[j]=t;

}

}

}

}

void move\_DVD\_stack(struct Stack\* s, struct Disks\* d){

if (strcmp(d->type,"DVD")==0){

push(s,d);

printf("\n%s is moved to the stack\n", d->mname);

}

}

void check\_2018(struct Stack\* s, struct Stack\* s1,struct Queue\* q){

struct Disks\* d=(struct Disks\*)malloc(sizeof(struct Disks));

for (int i=0;i<s->top;i++){

d=pop(s);

if (strcmp(d->type,"DVD")==0 && d->year==2018){

printf("\n%s is added to the queue\n",d->mname);

enqueue(q,d);

}

else{

push(s1,d);

}

}

}

int main(){

struct Stack\* s=(struct Stack\*)malloc(sizeof(struct Stack));

s->top=-1;

struct Stack\* s1=(struct Stack\*)malloc(sizeof(struct Stack));

s1->top=-1;

struct Queue\* q=(struct Queue\*)malloc(sizeof(struct Queue));

q->rear=-1;

q->front=-1;

struct Queue\* q2=(struct Queue\*)malloc(sizeof(struct Queue));

q2->rear=-1;

q2->front=-1;

int n;

printf("Enter n:");

scanf("%d",&n);

struct Disks d[n];

for (int i=0;i<n;i++){

printf("Enter the type(CD/DVD), ID, year, mname:\n");

scanf("%s %d %d %s",d[i].type, &d[i].ID, &d[i].year, d[i].mname);

}

for (int i=0;i<n;i++){

check\_year(q,&d[i]);

}

sort\_disks(d,n);

for (int i=0;i<n;i++){

move\_DVD\_stack(s,&d[i]);

}

check\_2018(s,s1,q2);

return 0;

}

