STACK::::::::------------------------------------------------------------------------------------------------------------------------

#include<bits/stdc++.h>

#define CAPACITY 3

int tack[CAPACITY];

using namespace std;

int top =-1;

void push(int x){

if(top<CAPACITY-1){

top=top+1;

tack[top]= x;

cout<<x<<endl;

}

else{

cout<<"exception no space\n";

}

}

int pop(){

if(top>=0){

int valu=tack[top];

top=top-1;

return valu;

}

cout<<"exception empty\n";

return -1;

}

int peak(){

if(top>=0){

return tack[top];

}

cout<<"exception empty\n";

return -1;

}

int main()

{

peak();

push(10);

push(11);

push(12);

pop();

push(145);

cout<<peak()<<endl;

return 0;

}

QUEUE: ------------------------------------------------------------------------------------------------------------------------------

#include<bits/stdc++.h>

#define CAPACITY 5

int data[CAPACITY];

using namespace std;

int font=0,rear=-1,totalitam=0;

bool isfull()

{

if(totalitam==CAPACITY)

{

return true;

}

return false;

}

bool isampty(){

if(totalitam==0)

{

return true;

}

return false;

}

void enqueue(int item)

{

if(isfull()){

cout<<"sorry the list is full\n";

return;

}

rear=(rear+1)%CAPACITY;

data[rear]=item;

totalitam++;

}

void printinfo(){

int i;

cout<<"queue:";

for(i=0;i<CAPACITY;i++){

cout<<data[i]<<"\t";

}

cout<<endl;

}

int dequeue()

{

if(isampty()){

cout<<"no data here\n";

return -1;

}

int fontitam = data[font];

data[fontitam] = -1;

font = (font+1)%CAPACITY;

totalitam--;

return data[fontitam];

}

int main()

{

dequeue();

enqueue(10);

enqueue(11);

enqueue(12);

enqueue(13);

enqueue(14);

printinfo();

enqueue(15);

dequeue();

dequeue();

printinfo();

enqueue(15);

enqueue(16);

printinfo();

return 0;

}

#include<bits/stdc++.h>

#define CAPACITY 5

int ourQueue[CAPACITY];

using namespace std;

int front=0,rear=-1,totalitam=0;

bool isfull()

{

if(totalitam==CAPACITY)

{

return true;

}

return false;

}

bool isampty(){

if(totalitam==0)

{

return true;

}

return false;

}

void enqueue(int item)

{

if(isfull()){

cout<<"sorry the list is full\n";

return;

}

rear=(rear+1)%CAPACITY;

ourQueue[rear]=item;

totalitam++;

}

void printinfo(){

int i;

cout<<"queue:";

for(i=0;i<CAPACITY;i++){

cout<<ourQueue[i]<<"\t";

}

cout<<endl;

}

int dequeue(){

if(isampty()){

printf("Sorry, the Queue is empty.\n");

return -1;

}

int frontItem = ourQueue[front];

ourQueue[front] = -1;

front = (front + 1) % CAPACITY;

totalitam--;

return frontItem;

}

int main()

{

dequeue();

enqueue(10);

enqueue(11);

enqueue(12);

enqueue(13);

enqueue(14);

printinfo();

enqueue(15);

dequeue();

dequeue();

printinfo();

enqueue(15);

enqueue(16);

printinfo();

return 0;

}

Link list--------------------------------------------------------------------------------------------------------------------------------

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

struct Node \*next;

};

int main()

{

struct Node \*a=NULL;

struct Node \*b=NULL;

struct Node \*c=NULL;

a=(struct Node\*)malloc(sizeof(struct Node));

b=(struct Node\*)malloc(sizeof(struct Node));

c=(struct Node\*)malloc(sizeof(struct Node));

a->data=10;

b->data=100;

c->data=1000;

a->next=b;

b->next=c;

c->next=NULL;

while(a!=NULL){

cout<<a->data<<endl;

a= a->next;

}

return 0;

}

Link list ----------------------------------------------------------------------------------------------------------------------------

#include <bits/stdc++.h>

using namespace std;

struct node \*creatalinklist(int ar[],int size);

struct node{

int data;

struct node \*naxt;

};

int main()

{

int a[]={1,5,6,20,40,80,10,20};

struct node \*head;

head=creatalinklist(a,8);

while(head!=NULL){

printf("%d ->",head->data);

head=head->naxt;

}

printf("NULL\n");

return 0;

}

struct node \*creatalinklist(int ar[],int size){

struct node \*head=NULL,\*temp=NULL,\*curr=NULL;

int i;

for(i=0;i<size;i++){

temp=(struct node\*)malloc(sizeof(struct node));

temp->data=ar[i];

temp->naxt=NULL;

if(head==NULL){

head=temp;

curr=temp;

}

else{

curr->naxt= temp;

curr=curr->naxt;

}

}

return head;

}

Find link list---------------------------------------------------------------------------------------------------------------------------

#include <bits/stdc++.h>

using namespace std;

struct node \*creatalinklist(int ar[],int size);

int sarcelink(struct node \*head,int value);

struct node{

int data;

struct node \*naxt;

};

int main()

{

int a[]={1,5,6,20,40,80,10,20};

struct node \*head;

head=creatalinklist(a,8);

printf("index :%d\n\n\n",sarcelink(head,1));

while(head!=NULL){

printf("%d ->",head->data);

head=head->naxt;

}

printf("NULL\n");

return 0;

}

int sarcelink(struct node \*head,int value){

int index=1;

while(head!=NULL){

if(head->data==value){

return index;

}

index++;

head=head->naxt;

}

return -1;

}

struct node \*creatalinklist(int ar[],int size){

struct node \*head=NULL,\*temp=NULL,\*curr=NULL;

int i;

for(i=0;i<size;i++){

temp=(struct node\*)malloc(sizeof(struct node));

temp->data=ar[i];

temp->naxt=NULL;

if(head==NULL){

head=temp;

curr=temp;

}

else{

curr->naxt= temp;

curr=curr->naxt;

}

}

return head;

}