/\*//DAY-9

1.Write a menu based program to implement a simple stack (static),

Implmenet functinos to add, delete and display the content of the stack

#include <stdio.h>

#include <stdlib.h>

#define ssize 5

void push(int a[],int \*top,int ele)

{

if(\*top==ssize-1)

{

printf("stack overflow\n");

return;

}

(\*top)++;

a[\*top]=ele;

}

void pop(int a[],int \*top)

{

int ele;

if(\*top==-1)

{

printf("stack Under flow");

return;

}

ele= a[\*top];

(\*top)--;

printf("The element deleted%d\t",ele);

}

void display(int a[],int top)

{

int i;

if(top==-1)

{

printf("stack Under flow");

return;

}

printf("Stack contents are\n");

for(i=0;i<=top;i++)

{

printf("%d",a[i]);

}

}

void main()

{

int a[ssize];

int top=-1;

int ele;

int choice;

int dele;

for(;;)

{

printf("Menu\t1:Push\t2:pop\t3:display\t4:exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("enter the ele to be inserted\n");

scanf("%d",&ele);

push(a,&top,ele);

break;

case 2: pop(a,&top);

break;

case 3: display(a,top);

break;

case 4: exit(0);

}

}

}

2.Write a menu based program to implement dynamic stack,

a stack that never gets filled( until stack segment is full)

Implmenet functinos to add, delete and display the content of the stack

Hint: start with intial capacity 1/2

use malloc, realloc

#include <stdio.h>

#include <stdlib.h>

struct stack

{

int key;

};

typedef struct stack element;

element\* stackFull(element \*stack,int \*capacity)

{

printf("Wait stack is getting doubled\n");

stack=(element\*)realloc(stack,2\*(\*capacity)\*sizeof(element));

(\*capacity)=(\*capacity)\*2;

return stack;

}

element\* push(element \*stack,int \*top,int item,int \*capcity)

{

if(\*top>=(\*capcity)-1)

{

stack=stackFull(stack,capcity);

}

(\*top)++;

(stack+\*top)->key=item;

return stack;

}

int pop(element \*stack,int \*top)

{

int item\_del;if(\*top==-1)

{

printf("stack underflow\n");

return -1;

}

item\_del=((stack+\*top))->key;

(\*top)--;

return item\_del;

}

void display(element \*stack,int top)

{

int i;

if(top==-1)

{

printf("stack underflow\n");

return ;

}

for(i=top;i>=0;i--)

{

printf("%d\t",(stack+i)->key);

}

printf("\n");

}

void main()

{

int top=-1,capacity=1,choice,item\_deleted,item;

element \*stack;

stack=(element\*)malloc(sizeof(element)\*capacity);

if(stack==NULL)

{

printf(" Moemory not suff...\n");

exit(0);

}

for(;;)

{

printf("1:push 2:pop 3:display 4:exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("Enter the element to be inserted\n");

scanf("%d",&item);

stack=push(stack,&top,item,&capacity);

break;

case 2: item\_deleted=pop(stack,&top); if(item\_deleted!=-1)

{

printf("Deleted item %d",item\_deleted);

}

break;

case 3:display(stack,top);

break;

case 4:exit(0);

}

}

}

3.Write a menu based program to implement a simple/ linear queue (queue),

Implmenet functinos to add, delete and display the content of the stack

discuss the limitation of this simple linear queue

#include <stdio.h>

#include <stdlib.h>

#define Q\_size 5

void insert(int q[],int item,int \*r)

{

if(\*r==Q\_size-1)

{

printf("Q full\n");

return;

}

\*r=\*r+1;

q[\*r]=item;

}

int delet(int q[],int \*f,int \*r)

{

if(\*f>\*r)

{

return -1;

}

return q[(\*f)++];

}

void display(int q[],int f,int r)

{

int i;

if(f>r)

{

printf("Q empty\n");

return;

}

for(i=f;i<=r;i++)

{

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

}

}

void main()

{

int f=0,r=-1,choice,q[10],item;

for(;;)

{

printf("1:insert 2:delete 3: dipslay 4:exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("Enter the element to be inser\n");

scanf("%d",&item);

insert(q,item,&r);

break;

case 2: item=delet(q,&f,&r);

if(item==-1)

printf("q empty\n");

else

printf("item\_deleted=%d",item);

break;

case 3: display(q,f,r);

break;

case 4:exit(0);

}

}

}

4.Demonstrate the working of stack using a book database example.

Make use of the function template given in the program and complete the code.

Analyse the following

a. parameter passing in structures

b. assigning strucure members with just assignment operators

#include <stdio.h>

#include<stdlib.h>

#define SIZE 20

#define TRUE 1

#define FALSE 0

struct book

{

int book\_id;

char title[10];

char author[10];

float price;

int no\_pages;

};

void push(struct book, int \*, struct book[]);

struct book pop(int \*, struct book []);

struct book peep(int , struct book []);

void display(int , struct book []);

int stackfull(int \*);

int stackempty(int \*);

int stackfull(int \*tos)

{

if( \*tos==SIZE)

return TRUE;

return FALSE;

}

int stackempty(int \*tos)

{

if(\*tos == -1)

return TRUE;

return FALSE;

}

void push(struct book ele, int \*tos, struct book stack[])

{

(\*tos)++;

stack[\*tos]=ele;

}

struct book pop(int \*tos, struct book stack[])

{

struct book e;

e=stack[\*tos];

\*tos--;

return e;

}

struct book peep(int tos,struct book stack[])

{

return stack[tos];

}

void display(int tos, struct book stack[])

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Bppk\_Id\t Title\t Author\t Price\t No\_pages\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for (int i=tos; i>=0; i--)

{

printf("%d\t %s\t %s\t %.2f\t %d\n", stack[i].book\_id, stack[i].title, stack[i].author, stack[i].price, stack[i].no\_pages);

}

}

int main()

{

struct book stack[SIZE], ele;

int choice, item;

int top=-1;

struct book pop\_item, peep\_item;

for(;;)

{

printf("\tEnter 1 for push, 2 for pop 3 for pip and 4 for display 5 for exit\n");

scanf("%d", &choice);

switch(choice)

{

case 1: if(stackfull(&top))

{

printf("The Stack is full\n");

break;

}

printf("Enter the book id title, author name, price and number of pages in order\n");

scanf("%d%s%s%f%d", &ele.book\_id, ele.title, ele. author, &ele.price, &ele.no\_pages);

push(ele,&top,stack);

break;

case 2: if (stackempty(&top))

{

printf(" The stack is empty \n");

break;

}

pop\_item=pop(&top, stack);

printf("The detailes of popped record are\n");

printf("%d\t %s\t %s\t %.2f\t %d\n", pop\_item.book\_id, pop\_item.title, pop\_item.author, pop\_item.price, pop\_item.no\_pages);

break;

case 3: if (stackempty(&top))

{

printf(" The stack is empty \n");

break;

}

peep\_item=peep(top, stack);

printf("The item at the top of the stack is\n");

printf("%d\t %s\t %s\t %.2f\t %d\n", peep\_item.book\_id, peep\_item.title, peep\_item.author, peep\_item.price, peep\_item.no\_pages);

break;

case 4 : if (stackempty(&top))

{

printf(" The stack is empty \n");

break;

}

printf("The content of the stack are \n");

display(top, stack);

break;

case 5:exit(0);

default: printf("Enter a valid choice\n");

}

}

return 0;

}

\*/