**Exercise 4: StackADT and its Applications**

**stackADT.h:**

#include<stdio.h>

struct stack

{

int max,top,data[100];

};

void display(struct stack);

void pop(struct stack \*);

void push(struct stack \*,int x);

int isEmpty(struct stack \*);

int isFull(struct stack \*);

void initstack(struct stack \*s,int max);

int top(struct stack \*);

//int game(struct stack \*, struct stack \*);

**stackimpl.h:**

#include "stackADT.h"

void initstack(struct stack \*s,int max)

{

s->top=-1;

s->max=max;

}

int isFull(struct stack \*s)

{

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

return 1;

else

return 0;

}

int isEmpty(struct stack \*s)

{

if(s->top==-1)

return 1;

else

return 0;

}

void push(struct stack \*s,int x)

{

if(isFull(s))

{printf("\nStack is Full!\n");}

else

{

s->top++;

s->data[s->top]=x;

}

}

void pop(struct stack \*s)

{

if(isEmpty(s))

{printf("\nStack is Empty!\n");}

else

{

s->top--;

}

}

void display(struct stack s)

{

if(isEmpty(&s))

{printf("\nStack is empty!\n");}

else

{

printf("Stack:\n");

int i=s.top;

while(i>=0)

{

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

}

}

}

int top(struct stack \*s)

{

if(isEmpty(s))

{

//printf("Empty Stack!");

return -1;

}

else

return s->data[s->top];

}

**stackappl.c:**

#include "stackimpl.h"

#include <stdlib.h>

int game(struct stack \*A, struct stack \*B,int max\_sum);

int main()

{

int ch,max;

struct stack s,q,p;

printf("\nEnter size of stack: ");

scanf("%d",&max);

initstack(&p,max);

printf("\n----------------------------------------\n\nMENU:\n\n1.Push\n2.Pop\n3.Display\n4.Top\n5.Play Game\n6.Exit\n");

do{

int ele,sum;

//printf("\n----------------------------------------\n\nMENU:\n\n1.push\n2.pop\n3.display\n4.Game\n5.Exit\nEnter your choice:");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("Enter a element to insert: ");

scanf("%d",&ele);

push(&p,ele);

display(p);

break;

case 2: printf("Element is popped!,\nPopped element: %d\n",top(&p));

pop(&p);

//display(p);

break;

case 3:display(p);

break;

case 4:printf("\nTop of the Stack: %d",top(&p));

break;

case 5:initstack(&s,10);

initstack(&q,10);

push(&s,1);

push(&s,6);

push(&s,4);

push(&s,2);

push(&s,4);

push(&q,5);

push(&q,8);

push(&q,1);

push(&q,2);

printf("\nGAME:\nEnter the Sum: ");

scanf("%d",&sum);

int count=game(&s,&q,sum);

printf("\nCount: %d\n",count);

break;

case 6:printf("\nExiting...\n");

break;

default:printf("\nInavlid input!\n");

}

}while(ch!=6);

}

int game(struct stack \*A, struct stack \*B,int max\_sum)

{

int sum=0,n=0,ch;

while(!isEmpty(A)&&!isEmpty(B)&&sum<max\_sum)

{

printf("\nStack1\tStack2\tSum\n%d\t%d\t%d\n",top(A),top(B),sum);

//printf("\nEnter which stack to remove from: ");

//scanf("%d",&ch);

ch=rand()%2+1;

if(ch==1)

{

sum+=top(A);

pop(A);

n++;

}

else if(ch==2)

{

sum+=top(B);

pop(B);

n++;

}

else

printf("\nInvalid Input!\n");

}

while(!isEmpty(A)&&sum<max\_sum)

{

printf("\nStack1\tStack2\tSum\n%d\t%d\t%d\n",top(A),top(B),sum);

sum+=top(A);

pop(A);

n++;

}

while(!isEmpty(B)&&sum<max\_sum)

{

printf("\nStack1\tStack2\tSum\n%d\t%d\t%d\n",top(A),top(B),sum);

sum+=top(B);

pop(B);

n++;

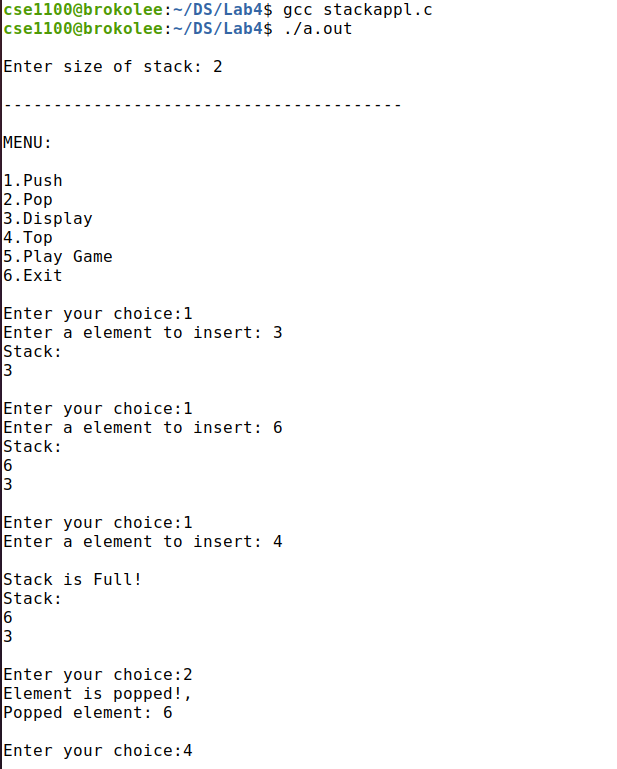
}

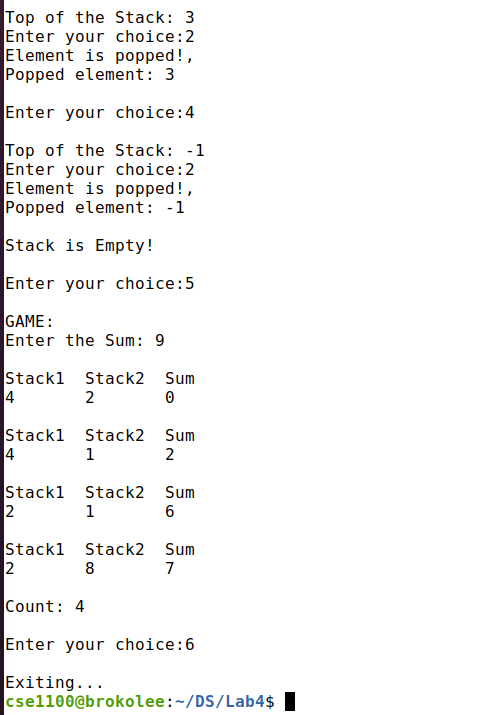
return n;

}

**Sample I/O:**

**(new one):**

****

****

**(OLD ONE):**

