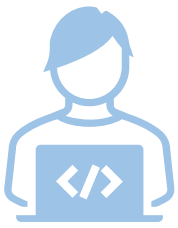
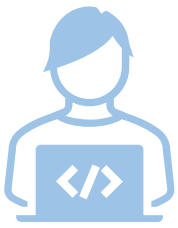
Stack .h

```
#ifndef STACK1_H_INCLUDED
#define STACK1_H_INCLUDED
#define Max 4
typedef int type;
typedef struct{
type top;
type arrry[Max];
}stack;
void creatstack(stack *s);
int stackempty(stack s);
int stackfull(stack s);
void push(type item,stack *s);
type pop(stack *s);
void print(stack s);
#endif // STACK_H_INCLUDED
```



Stack.c

```
#include "stack1.h"
# include <stdlib.h>
void createstack (stack *s)
{  s->top=-1;}
int stackempty(stack s)
{return(s.top==-1);}
int  stackfull(stack s)
{ return(s.top==Max-1);}
void push(type item,stack *s)
{s->arrry[++s->top]=item;}
type pop(stack *s)
{ type item;
item=s->arrry[s->top--];
return item;}
void printstack(stack s)
{for(int i=s.top;i>=0;i--)
{ printf("%d\n",s.arrry[i]); }  }
```



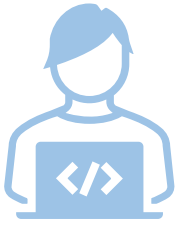
Q.1

Write a function that return the max element in a stack (user level)

Solution

In main.c add following function above main function:-

```
int getmax(stack *s){
   int item;
   int maxnum=0;
   stack temp;
   createstack (&temp);
    while(!stackempty(*s))
              {
               item=pop(s);
               push(item,&temp);
                 if(maxnum<item)
                 maxnum=item;
              }
   while(!stackempty(temp))
            {
              item=pop(&temp);
               push(item,s);
             }
           return maxnum;
     }
```



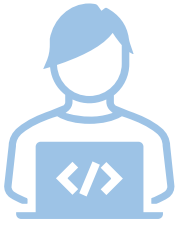
Q.2

Write a function that return the max element in a stack (implementation)

Solution

In stack.c add following function :-

```
int getmax(stack *s)
{
   int max=0;
   for(int i=s->top;i>=0;i--)
   {
      if(max<s->arrry[i])
        max=s->arrry[i];
   }
  return max;
}
```



Q.3

Write a C Function , You will Pass to it a Stack that Split it into two stacks , Stack for Even Numbers and another one for Odd Numbers , and return the Stack without Changing in the Original's Values (userlevel)

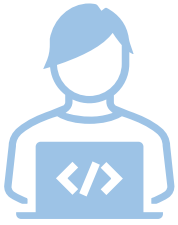
Function Header:

void SplitTwoEvenAndOdd (Stack *Original , Stack *Even , Stack *Odd)

Solution

In main.c add following function above main function:-

```
void spiltevenAndodd(stack *orignalstack,stack *evennum,stack *oddnum)
{
   int item;
   stack temp;
   creatstack(&temp);
   while(!stackempty(*orignalstack))
   {
       item=pop(orignalstack);
       push(item,&temp);
           if (item%2==0)
               push(item,evennum);
            else
               push(item,oddnum);
   }
      while(!stackempty(temp))
```



```
{ item=pop(&temp); push(item,orignalstack); } }
```

Q.4

Write a C Function , You will Pass to it a Stack that Split it into two stacks , Stack for postive Numbers and another one for negative Numbers , and return the Stack without Changing in the Original's Values (userlevel)

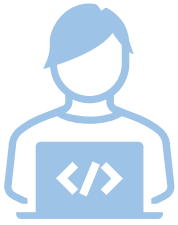
Function Header:

void SplitTwopostiveAndnegative (Stack *Original , Stack *postive , Stack *negative)

Solution

In main.c add following function above main function:-

```
void spiltpostiveAndnegative(stack *originalstack,stack *postive,stack *negative)
{
   int item;
   stack temp;
   creatstack(&temp);
   while(!stackempty(*originalstack))
     {
      item=pop(originalstack);
      push(item,&temp);
      if (item>=0)
           push(item,postive);
      else
            push(item,negative);
```



```
    }
        while(!stackempty(temp))
    {
    item=pop(&temp);
    push(item,originalstack);
    }
}
```

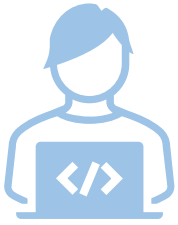
Q.5

Write a C Function which you will pass to it Stack and the element you search for , without changing the Stack Elements, You will enter an element and the function will return 1 if the element is exists and 0 otherwise (user level)

Solution

In main.c add following function above main function:-

```
int getelement(stack *s,int element)
{
    int item,check;
    stack temp;
    creatstack(&temp);
    while(!stackempty(*s))
      {
        item=pop(s);
        push(item,&temp);
        if(element==item)
```



```
        {
            check=1; break;  }
        else
            check=0;
    }
        while(!stackempty(temp))
    {
    item=pop(&temp);
    push(item,s);
    }
    return check;
}
```

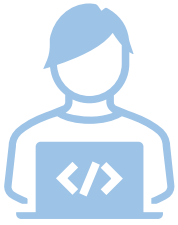
Q.6

Write a C Function which you will pass to it Stack and the element you search for , without changing the Stack Elements, You will enter an element and the function will return 1 if the element is exists and 0 otherwise (implementation)

Solution

In stack.c add following function :-

```
int checkelement(stack *s, int element)
{
    int check;
     for(int i=s->top ; i>=0 ; i--)
    {
```



```
        if(element==s->arrry[i])
         {
            check=1; break; }
         else
           check = 0 ;
    }
    return check;
}
```

By : Mido

Revised By : 3BS