Lab 6 Documentation

11.c

```
//strategy: we will use stack for to evaluate postfix as we need to push and pop form the same end
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//function to convert string into integer
int stoi(char *str)
{
    int x;
    sscanf(str, "%d", &x);
    return x;
}

//stack structure
typedef struct stack{
    int value;//because we will preform task when we get operator, ie stores only integer
    struct stack* next;
}stack;

//headPointer intializes to null characters and size to zero
stack* head=NULL;
int size=0;
//function to insert element at the beginning
int push(int i)
{
        stack* newNode=(stack*)malloc(sizeof(stack));
        newNode->value=i;
        newNode->next=head;
        head=newNode;
        size++;
        return 0;
}
//function to dlt element from the beginning
int pop()
```

```
{
	if(head==NULL)
	return -1;

   stack* temp=head;
   head=temp->next;
   int deletedElement=temp->value;
   free(temp);
   size--;
      return deletedElement;
}
//function to get the top element of the stack
int peekFront()
{
	if(head==NULL)
	return -1;
	return head->value;
}
```

```
int main()
{
	char a[10];
	char line[128];
//scan till end ie till we get null
	while(fgets(line, sizeof line, stdin) != NULL )
	{	sscanf(line,"%s",a);
		//operation will take place only if we get top two valid integer to get a valid integer result
		if(a[0]=='+')
		{
			int c=pop();
			int d=pop();
			//invalid otherwise, return
			if(c==-1||d==-1)
			{
				printf("invalid\n");
				return 0;

			}
```

```
        //if operator is + then add top two integers and push result on top of the stack
        push(d+c);
}

else if(a[0]=='/')
{
        int c=pop();
        int d=pop();
        if(c==-1||d==-1)
        {
                printf("invalid\n");
                return 0;

        }
        //if devisor is 0 print division by zero and return
        if(c==0)
        {
                printf("division by 0\n");
                return 0;
        }
        //if operator is / then devide second last by top and push result on top of the stack
        push(d/c);
}

else if(a[0]=='-')
{
        int c=pop();
        int d=pop();
        if(c==-1||d==-1)
        {
                printf("invalid\n");
                return 0;

        }
        //if operator is - then substract second last from top and push result on top of the stack
        push(d-c);
}
```

```
else if(a[0]=='*')
{
	int c=pop();
	int d=pop();
	if(c==-1||d==-1)
	{
		printf("invalid\n");
		return 0;

	}
	//if operator is * then multiply both and push result on top of the stack
	push(d*c);
}
//else means a is a integer string
//so push integer of the string on the top
else
{
	push(stoi(a));
}
	}
	//after completion of the loop, we must have size ==1
	//otherwise invalid because we have no more any charcter to perform the operation
	if(size>1||size==0)
	{
		printf("invalid\n");
		return 0;
	}
	//if size ==1 , means that will be our answer
	if(size==1)
	{
		printf("%d\n",pop());
		return 0;
	}
//congratulations! you have made it..
}
```

21.c

```
//strategy:using a char stack structure to store and print the answer string
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//function to convert string into integer
int stoi(char *str)
{
    int x;
    sscanf(str, "%d", &x);
    return x;
}
//stack structure
typedef struct stack{
    char value;
    struct stack* next;
}stack;
//headPointer intializes to null characters and size to zero
stack* head=NULL;
int size=0;
//function to insert char element at the beginning
void push(char i)
{
        stack* newNode=(stack*)malloc(sizeof(stack));
        newNode->value=i;
        newNode->next=head;
        head=newNode;
        size++;

}
//function to dlt element from the beginning
char pop()
{
        if(head==NULL)
        return '\0';
```

```
    stack* temp=head;
    head=temp->next;
    char deletedElement=temp->value;
    free(temp);
    size--;
        return deletedElement;
}
//function to get the top element of the stack
char peekFront()
{
        if(head==NULL)
        return '\0';

        return head->value;
}

int main()
{       int i=0;
        char a[128];
scanf("%s",a);
//check untill we got null character
while(a[i]!='\0' )
{
        if(a[i]=='+'||a[i]=='-'||a[i]=='/'||a[i]=='*')
        {       while(
                if(peekFront()=='\0'||peekFront()=='(')
                {
                        push(a[i]);
                }
                if((a[i]=='*'||a[i]=='/')&&(peekFront()=='+'||a[i]=='-'))
                {
                        push(a[i]);
                }
                if((a[i]=='+'||a[i]=='-')&&(peekFront()=='*'||a[i]=='/'))
                {
                        printf("%c",pop());

                }
```

```
                if((a[i]=='+'&&peekFront()=='-')||(a[i]=='*'&&peekFront()=='/')||
(a[i]=='/'&&peekFront()=='*')||(a[i]=='-'&&peekFront()=='+'))
                {
                        printf("%c",pop());
                        push(a[i]);
                }



        }


        else if(a[i]=='(')
        {
                push(a[i]);
        }

        else if(a[i]==')')
        {       char c=pop();
                while(c!='(')
                {
                printf("%c",c);
                c=pop();
                }

        }

        else
        {
                printf("%c",a[i]);
        }
i++;

}

while(head!=NULL||pop()!='\0')
{
```

```
                printf("%c",pop());
        }
printf("\n");
        return 0;

}
```