Documentation

11.c

strategy: we will keep track of count of negative and non negative numbers in stack, push is straight forward but we have to do little hard work to implement pop as shifting each ele to left till we find the appropriate number to pop is tricky...

```
#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;
}

//int size=0;
//intially stack is empty
//positive is the number of positive ele in stack
int positive =0;
//positive is the number of positive ele in stack
int negative =0;



int push(int a[], int i,int n, int nn)
{      //if stack is full, either  positive or negative is overflowing
return -1
    if((i>=0&&positive==n)||(i<0&&negative==nn))
    return -1;

    //otherwise insert it at a[size] and incr the negative by one if
negative else incr positive
    if(i<0&&negative<nn)
```

```
   {
      a[positive+negative]=i;
      negative++;

   }
   if(i>=0&&positive<n)
   {
      a[positive+negative]=i;
      positive++;
   }
   return 0;


}

int popp(int a[])
{   //if there is no non negative number then there is no ele to
return
      if(positive==0)
      return -1;
      int i=positive+negative-1;
//else shift each ele to left starting from top till we find a non
negative number , at last decr positive by one
      while(a[i]<0&&i>=0)
      {
            i--;
      }
      int ans=a[i];
      while(i<positive+negative-1)
      {
            a[i]=a[i+1];
      i++;
      }
      positive--;
  //returning the topmost non negative element
      return ans;
}
```

```
int popn(int a[])
{    //if there is no negative number then there is no ele to return
        if(negative<=0)
        return 0;
        int i=positive+negative-1;
//else shift each ele to left starting from top till we find a negative
number , at last decr negative by one
        while(a[i]>=0&&i>=0)
        {
                i--;
        }
        int ans=a[i];
        while(i<positive+negative-1)
        {
                a[i]=a[i+1];
                i++;
        }
        negative--;
    //returning the topmost negative element
        return ans ;
}



void printp(int a[])
{   //printing ele which is non negative starting from top of stack if
there exist at least one non negative element
    if(positive>0)
    {  int i=positive +negative-1;
        while(i>=0)
        {    if(a[i]>=0)
            printf("%d\n",a[i]);
                i--;
        }
    }
}
```

```
void printn(int a[])
{  //printing ele which is non negative starting from top of stack if
there exist at least one non negative element
   if(negative>0)
   {  int i=positive +negative-1;
      while(i>=0)
      {    if(a[i]<0)
         printf("%d\n",a[i]);
            i--;
      }
   }
}

int main (int argc, char **argv)
{
   char line[128];
   char v1[15];
   char v2[15];
   char v3[15];

   int *A = NULL;
   int ret;
   int lineNo = 0;
   int n;
   int nn;

   while (fgets(line, sizeof line, stdin) != NULL )
   {
      sscanf(line, "%s %s %s", v1, v2, v3);
      lineNo++;

      if(lineNo == 1)
      {     n=stoi(v1);
         continue;
      }
      if(lineNo == 2)
      {     nn=stoi(v1);
//intializing array with size of total element
```

```
            A = (int*) malloc(sizeof(int)* (nn+n));
            continue;
        }

        if(strcmp(v1,"PSH") == 0)
        {
            ret = push(A, stoi(v2),n,nn);
            if(ret < 0)
                printf("%d\n", -1);
        }
        else if(strcmp(v1,"POPN") == 0)
        {
            ret = popn(A);
            printf("%d\n", ret);
        }
        else if(strcmp(v1,"POPP") == 0)
        {
            ret = popp(A);
            printf("%d\n", ret);
        }
        else if(strcmp(v1,"PRTN") == 0)
        {
            printn(A);
        }
        else if(strcmp(v1,"PRTP") == 0)
        {
            printp(A);
        }

        else
        {
            printf("INVALID\n");
        }
    }

    if(A)
        free(A);
    return 0;
}
```

21.c

strategy: making a structure which contains a integer and a pointer variable to next element

```
#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;
}

//defn of linked list named as stack
typedef struct stack{
    char value;
    struct stack* next;
}stack;

//the pointer variable to 1st ele of the stack
stack *head;
//global variable to keep track of the size of stack
int size=0;

int isEmpty()
{   //if head==null, return 1
    if(head==NULL)
    return 1;
        //otherwise return 0
        return 0;
}


//printing the size of stack
void Size()
{
        printf("%d\n",size);
}


int push(char i)
{       //creating a stack type new node and inserting at head and
incr the size
        stack* newNode=(stack*)malloc(sizeof(stack));
        newNode->value=i;
        newNode->next=head;
//making head now to point at newly made node
        head=newNode;
        size++;
        return 0;
}

char pop()
{   //if stack is empty,there is no ele to dlt, so return null character
        if(isEmpty()==1)
        return '\0';
    //otherwise returning the element at head and decr the size

    stack* temp=head;
    head=temp->next;
    char deletedElement=temp->value;
// deleting the temp now
    free(temp);
    size--;
        return deletedElement;
}

char peekFront()
```

```
{   //if stack is empty,there is no ele, so return -1
       if(isEmpty()==1)
       return -1;
    // otherwise return the value of head pointer
       return head->value;
}

void print()
{   //if stack is not empty,printing the elements in the the reverse
order in which they are  inserted ie as we inserted at head we will
also start printing from made
    if(isEmpty()==0)
    {   stack* temp;
        temp=head;
   //stop printing when there is no ele next ie when temp will be a
null pointer
        while(temp!=NULL)
        {
           printf("%c\n",temp->value);
           temp=temp->next;
        }
    }
}

int main (int argc, char **argv)
{
    char line[128];
    char v1[15];
    char v2[15];
    char v3[15];

    //int *A = NULL;
    int ret;
    int lineNo = 0;
    int n;
    head=NULL;
    while (fgets(line, sizeof line, stdin) != NULL )
    {
```

```
sscanf(line, "%s %s %s", v1, v2, v3);
lineNo++;

if(strcmp(v1,"PSH") == 0)
{
   push(v2[0]);

}
else if(strcmp(v1,"POP") == 0)
{
   ret = pop();
   if(ret=='\0')
   printf("0");
   printf("%c\n", ret);
}
else if(strcmp(v1,"TOP") == 0)
{
   ret = peekFront();
   if(ret=='\0')
   printf("0");
   printf("%c\n", ret);
}
else if(strcmp(v1,"PRT") == 0)
{
   print();
}
else if(strcmp(v1,"SZE") == 0)
{
    Size();

}
else if(strcmp(v1,"EMP") == 0)
{
   ret = isEmpty();
   printf("%d\n", ret);
}

else
```

```
        {
           printf("INVALID\n");
        }
    }


    return 0;
}
```

22.c
strategy: implentation using stack,

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//defn of structure
typedef struct stack{
    char value;
    struct stack* next;
}stack;

int size=0;
stack *head;

int push(char i)
{
        stack* newNode=(stack*)malloc(sizeof(stack));
        newNode->value=i;
        newNode->next=head;
        head=newNode;
        size++;
        return 0;
}
```

```
char pop()
{
       if(head==NULL)
       return '\0';

    stack* temp=head;
    head=temp->next;
    char deletedElement=temp->value;
    free(temp);
    size--;
       return deletedElement;
}

char peekFront()
{
       if(head==NULL)
       return '\0';

       return head->value;
}


int main (int argc, char **argv)
{
head=NULL;
 char a[10000];
//scanning a string of character input
 scanf("%s",a);
int i=0;
//looping till we don’t reach to null character ie end of the string
while(a[i]!='\0')
{
//if current ele is right bracket then top ele must be right bracket of
same type else the input is not balanced, return 0
//if  ele is right bracket of same we will pop the top as we have
found it’s matching
```

```
if(a[i]=='}')
{
        if(peekFront()=='{')
        pop();
        else
        {
        printf("0\n");
        return 0;
        }

}
else if(a[i]==']')
{
        if(peekFront()=='[')
        pop();
        else
        {
        printf("0\n");
        return 0;
        }

}
else if(a[i]==')')
{
        if(peekFront()=='(')
        pop();
        else
        {
        printf("0\n");
        return 0;
        }

}
//if bracket is left bracket of any type we will only push it in stack
else push(a[i]);
//we will then move to next ele of array.
i++;
```

```
}
// if loop is cmplt ie we have gone through all ele
// if it’s balanced we have popped all the ele
//so head will be null
if(head==NULL)
printf("1\n");
//otherwise there are still bracket which haven’t found any pair\
//ie input is unbalanced
else
printf("0\n");



return 0;

}
```