LIFO

FIFO

c.png

Unepilepeutetreutiliserpourrepresentuninterpreteurparexemplesil'onveuxverifiersil'expressionestcorrectcadunepar
parenthesefermante.
l]− >
expressioncorrect.
− >
expressionincorrect.

```
- void initpile(void);
- void empiler (<type> val);
- void depiler (<type> *val);
- int pilevide(void); /*delivre vrais si la pile est vide , faux sinon */
- void sommetpile(<type> *val); /* renvoie la valeur du sommet*/
```

```
#define tmax ...
<type> pile[tmax] ;
<type> sommet;

void initpile(void){
  sommet  = -1;
}

void empiler (<type> val){
    if(sommet < tmax-1){
      sommet++;
      pile[sommet]=val;
    }
}

void depiler (<type> *val){
    if(sommet > -1 ){ /*avant d'enlever un element verifier si la pile n'est pas vide*/
      *val = pile[sommet];
      sommet --;
    }
}

int pilevide(void) {    /*delivre vrais si la pile est vide , faux sinon */
  return (sommet == -1);
}

void sommetpile(<type> *val) { /* renvoie la valeur du sommet*/
    *val=pile[sommet];
}
typedef struct element{
  <type> val;
  struct element *suivant;
} t_element;

t_element * pile;

void initpile(void){
  pile=NULL;
}

int pilevide(void){
  return(pile==NULL);
}

void empiler(<type> val){
  t_element *nouv=malloc(sizeof(t_element));

  nouv->val=val;
  nouv->suivant=pile;
  pile=nouv;
}

void depiler(<type> *val){
  t_element * sommet ;
  if(!pilevide()){    /*avant d'enlever un element verifier si la pile n'est pas vide*/
      *val=pile->val;
      pile=sommet->suivant;
      free(sommet);
  }
}

void sommetpile(<type> *val){
  if(pile!=NULL){
```

```
          queue=0;
        }
        else{
          queue++;
        }
      file[queue]=val;
      nb_val++;
    }
}

void retirer (<type> *val){ /*retrait en tete de file */
  if(nb_val > 0){
    *val=file[tete];
    nb_vall--;

    if(tete==tmax-1){
      tete=0;
    }
    else{
      tete++;
    }
  }
}

int filevide(void){
  return(nb_val==0);
}

int filepleine(void){
  return(nb_val==tmax);
}
```