DS Algorithmique - 3IFA - mardi 11 octobre 2022

Contraintes

Les solutions au DS sont validées via la plateforme domjudge. Chaque élève a son propre login et son propre mot de passe. Vous pouvez envoyer plusieurs fois une solution pour un même problème sans aucune pénalité.

Vous avez le droit aux supports des cours, TD, TP et aussi à des ressources Internet, mais vous n'avez pas le droit de communiquer avec d'autres personnes. La DSI sauvegarde tout le trafic TCP/IP pendant le DS. Vous pouvez utiliser seulement le protocole http pour accéder aux différents sites Web. Une connexion à un outil de messagerie, réseau social etc invalide automatiquement votre participation au DS avec les conséquences prévues par le règlement des études.

Vous pouvez utiliser le langage C ou le C++, mais sans utiliser la STL.

Problème 2 : pile implémentée par une liste chainée (4 points)

Vous devez compléter le code de la fonction empiler() ci-dessous afin qu'il fonctionne correctement pour l'implémentation d'une pile avec une liste chaînée. Ce code compile dans sa forme actuelle mais ne fait pas les opérations d'insertion.
Toutes les opérations d'entrées-sorties sont déjà implémentées, vous n'avez pas à y toucher.
Rappel : pour une pile implémentée en liste chainée, toutes les opérations se font sur la tête de la liste chaînée (empiler = ajouter en tête, dépiler = supprimer en tête).

Code à compléter

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Cellule {
   int valeur;
   struct Cellule * suivant;
} Cellule;

typedef struct Pile {
   Cellule * tete;
} Pile;

void empiler(Pile * pile, int val) {
   /* votre code ici */
}

int depiler(Pile * pile, int * res) {
   Cellule * suivant;
   if (pile->tete==NULL) {
      return 0;
   }
   *res = pile->tete->valeur;
   suivant = pile->tete->suivant;
   free(pile->tete);
   pile->tete = suivant;
   return 1;
```

```
}

int vide(Pile * pile) {
   return pile->tete==NULL;
}

Pile* init(void) {
   Pile *  pile;
   pile = (Pile*) malloc(sizeof(Pile));
   pile->tete = NULL;
   return pile;
}

void destroy(Pile * pile) {
   Cellule * todestroy;
   while(pile->tete) {
      todestroy = pile->tete;
      pile->tete = pile->tete->suivant;
      free(todestroy);
   }
   free(pile);
}

int main(void)
{
   char lecture[100];

   Pile *  pile;
   pile = init();
   int val;

   fscanf(stdin,"%99s",lecture);
   while (strcmp(lecture,"bye")!=0)
   {
      if (strcmp(lecture,"push")==0)
      {
         fscanf(stdin,"%99s",lecture);
         val = strtol(lecture,NULL,10);
         empiler(pile,val);
      }
      else if (strcmp(lecture,"pop")==0)
      {
         if (depiler(pile,&val))
            printf("%d\r\n",val);
      }
      else if (strcmp(lecture,"vide")==0)
      {
            printf("%s\r\n",vide(pile)?"O":"N");
      }
      fscanf(stdin,"%99s",lecture);
   }
   destroy(pile);
   return 0;
}
```