アルゴリズムとデータ構造 授業中練習問題6

次のプログラムは「動的スタックの実現例」である．このプログラムに関して，以下の問いに答えなさい．さらに，このプログラムを入力し，自分のパソコンでコンパイル，実行できることを確認してください．なお，プログラムの日本語部分は，英語，ローマ字に変更してかまいません

```
#include  <stdio.h>
#include  <string.h>
#include  <stdlib.h>
#define   String_Max  80

/*--- 文字列型スタックを実現する構造体 ---*/
typedef struct{
  int  max;   /* スタックの容量 */
  int  ptr;   /* スタックポインタ */
  char **stk; /* スタック本体(char* へのポインタ配列 )*/
} StringsStack;

/*--- スタックの初期化 ---*/
int Initialize(StringsStack *s, int max){
 s->ptr = 0;
 if ((s->stk = calloc(max, sizeof(char *))) == NULL) {
    s->max = 0;  /* char* の配列の確保に失敗 */
    return -1;
  }
  /* char* の配列の確保に成功 */
  s->max = max;
  return 0;
}

/*--- スタックの後始末 ---*/
void Terminate(StringsStack *s){
  if (s->stk != NULL){
    while( --s->ptr >= 0)
       free(s->stk[s->ptr]); /* 動的な文字列保存用配列を解放 */
    free(s->stk);
  }
  s->max = s->ptr = 0;
}

/*--- スタックにデータをプッシュ ---*/
int Push(StringsStack *s, char *x){
  if (s->ptr >= s->max) return -1; /* スタック満杯 */
  if ((s->stk[s->ptr] = calloc(strlen(x)+1, sizeof(char))) == NULL)
    /* データをコピーするための動的な文字列保存用配列を確保することに失敗 */
    return -1;
  strcpy(s->stk[s->ptr], x);
  s->ptr++;
  return 0;
}

/*--- スタックからデータをポップ ---*/
int Pop(StringsStack *s, char *x){
  if (s->ptr <= 0) return -1; /* スタックは空 */
  s->ptr--;
  strcpy(x, s->stk[s->ptr]);
  free(s->stk[s->ptr]); /* ポップしたので，動的な文字列保存用配列を解放 */
  return 0;
}
/*--- スタックからデータをピーク ---*/
```

```
int Peek(StringsStack *s, char *x){
  if (s->ptr <= 0) return -1;
  strcpy(x, s->stk[s->ptr - 1]);
  return 0;
}

/*--- スタックの容量 ---*/
int Capacity(const StringsStack *s){
  return s->max;
}

/*--- スタックに積まれているデータ数 ---*/
int Size(const StringsStack *s){
  return s->ptr;
}

/*--- スタックの全データの表示 ---*/
void Print(const StringsStack *s){
  int i;

  for(i = 0; i < s->ptr; i++)
    printf("%s¥n", s->stk[i]);
}

int main(void){
  StringsStack s;
  int max;

  printf("スタックの大きさを入力してください");
  scanf("%d", &max);

  if (Initialize(&s, max)==-1){
     puts("スタックの生成に失敗しました。");
    return 1;
  }

  while (1) {
    int  menu;
    char x[String_Max];

    printf("現在のデータ数：%d/%d¥n",Size(&s), Capacity(&s));
    printf("(1)プッシュ (2)ポップ (3)ピーク (4)表示 (0)終了：");
    scanf("%d", &menu);
    ―――――――――――――――― ①
    if (menu == 0) break;

    switch (menu) {
     case 1: /* プッシュ */
       printf("プッシュする文字列：");
       scanf("%s", x);
       if (Push(&s, x) == -1)
          puts("¥a エラー：プッシュに失敗しました. ¥n");
       break;
     case 2: /* ポップ */
       if (Pop(&s, x) == -1)
          puts("¥a エラー：ポップに失敗しました. ¥n");
       else{
           printf("ポップした文字列は%s, ", x);
       }
       break;
     case 3: /* ピーク */
```

```
        if (Peek(&s, x) == -1)
          puts("¥a エラー：ピークに失敗しました. ¥n");
        else{
          printf("ピークした文字列は%s, ", x);
        }
        break;
      case 4: /* 表示 */
        Print(&s);
        break;
      }
    }
  Terminate(&s);
  return 0;
}
```

1）このプログラムの動作直後に－1 を入力すると,「スタックの生成に失敗しました。」とのメッセージを表示して，プログラムが終了しました.「スタックの生成に失敗しました。」のメッセージを表示し，プログラムが終了することになった原因を説明しなさい.

このプログラムが動作直後に入力を求める値は，スタックの大きさである．従って，－1 は，スタックの大きさと解釈され，Initialize()関数の中で，スタックの実体である配列を確保するために利用されるが【calloc(-1, sizeof(char *))】，大きさが負の配列は確保できないため，配列の確保に失敗する．そのため，Initialize()の関数の値が-1 となり，「スタックの生成に失敗しました。」とのメッセージを表示して，プログラムは終了する.

2）このプログラムの動作直後に 0 を入力すると，プログラムは終了することなく動作しつづけ，下線部①の部分で入力待ちとなりました．このとき，次の問に答えなさい.

(ア) ここでポップを指示した場合，どのようなことが起こるか説明しなさい.

もし，大きさ 0 の配列が確保できたとしても，スタックにはデータが積まれていないため，ポップできず，エラーを知らせる音が鳴り，「エラー：ポップに失敗しました。」とのメッセージが表示される.

(イ) ここで，スタックに文字列" Maki"をプッシュすることを指示した場合，どのようなことが起こるか説明しなさい.

もし，大きさ 0 の配列が確保できたとしても，Initialize ()関数の中で s->ptr と s->max の値が共に 0 と設定されるので，スタックにデータを積もうとしたとき，Push()関数の if (s->ptr >= s->max) return -1 が成立する．従って，プッシュしようとする文字列" Maki"は，実際にはスタックにはプッシュされず，エラーを知らせる音が鳴り，「エラー：プッシュに失敗しました。」とのメッセージが表示される.

3）このプログラムの動作直後に 7 を入力し，スタックに文字列，”Ant”, "Duck", "Eagle", ”Pika”の順で積んだ後，下線部①の部分で入力待ちとなりました．このとき，次の問に答えなさい．

(ア)　この状態で，スタックから文字列を連続していくつ取り出せますか．

スタックに 4 つ文字列が積まれているので，連続して 4 個（4 回）の文字列が取り出せる．

(イ)　この状態から，スタックに連続して文字列をいくつ積めますか．

スタックの大きさを 7 と指定したので，スタックには全部で 7 個分の文字列が積める．今，4 つの文字列を積んだので，スタックの空き領域は 3 である．従って，この状態から 3 つの文字列を積むことができる．

4）このプログラムに，パターンの文字列を指定すると，スタック上の文字列の中から入力したパターンを，スタックの頂上から底に向かって探索し, 最初に見つけたパターンを含む文字列のスタックポインタの値を返す関数 Search を追加してください．関数 Search は以下のような動作をするものとします．追加した関数 Search を答えなさい．

- int Search(StringsStack *s , char *x)は，スタックに積まれている文字列の中から，パターンの文字列(x の先頭からパターンが入っている)で指定された文字列を，スタックの頂上から底に向かって探索し, 最初に見つけたパターンを含む文字列のスタックポインタの値を戻り値として返すようにしてください．ただし，パターンが見つからなかった場合は，-1 を返すようにしてください．

  関数 bm_mactch は，練習問題 4 で用いたものであるので，ここでは省略する．

```
/*---  スタック中のパターンの数を数える  ---*/
int Seach(StringsStack *s, char *pat){
   int pos = s->ptr ;

   while(pos>0)
      if   (bm_match(pat, s->stk[--pos]) != NULL)   return pos;
   return -1;
}
```