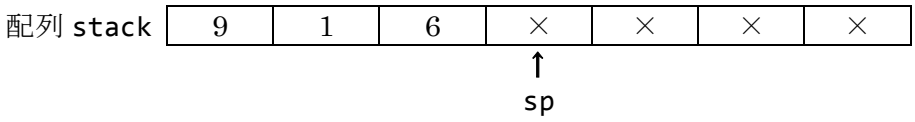
アルゴリズムとデータ構造　第８回演習　　年　　組　　番 氏名

1. **スタック**とは後入れ先出し（LIFO, 先入れ後出し FILO）のデータ構造であり，下記のプログラムはこれを配列で実現したものである。スタックにデータを格納する操作を**プッシュ**，スタックからデータを取り出す操作を**ポップ**という。適切に空欄を埋めてプログラムを完成させ，動作を確認せよ。

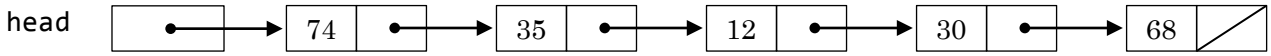



```
#include <stdio.h>
#include <stdlib.h>

#define N 100
int stack[N];  /* スタック */
int sp = 0;    /* スタックポインタ */

void push(int data)
{
    /* 空きがあるかチェック */
    if (          ) {

        printf("Stack is full.¥n");
        exit(1);
    }
    /* スタックにデータを「積む」 */
    stack[sp] = data;


}

int pop(void)
{
    /* データがあるかチェック */
    if (          ) {

        printf("Stack is empty.¥n");
        exit(1);
    }
    /* トップにあるデータを取り出す */


    return stack[sp];
}
```

```
void print_stack(void)
{
    int i;

    printf("[ ");
    for (i = 0; i < sp; i++)


    printf("]¥n");
}

/* スタックの動作確認プログラム */
int main(void)
{
    int data;
    int i;

    for (i = 0; i < 10; i++) {
        printf("Data %d: ", i);
        scanf("%d", &data);

        if (data > 0) {
            /* 入力が正数ならその数を格納 */
            push(data);
            printf("pushed: %d¥n", data);
        } else {
            /* 入力が負 or 0 なら取り出し */
            data = pop();
            printf("popped: %d¥n", data);
        }
        print_stack();
    }
    return 0;
}
```

2. 1.のプログラムにおいて，プッシュとポップはスタックのデータ数 n に対して$O(1)$のアルゴリズムである。もしスタックポインタ（sp）がないとどうなるか想定し，それが高速化に果たしている役割を考察せよ。

3. **連結リスト**は動的なデータ構造の一種であり，データを格納した**ノード**（またはセル）を**リンク**によって連結したものである。下記のプログラムは，1.と同様のスタックを単方向リストで実現したものである。適切に空欄を埋めてプログラムを完成させ，動作を確認せよ。また、このアルゴリズムではデータ数 n に対するプッシュとポップの計算量は$O(1)$になることを理解せよ。




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

/* リスト要素（ノード）の構造体の定義 */
struct node {
    int data;            /* 格納データ */

                *next; /* 次へのリンク */
};

/* リストの先頭ノードを指すポインタ */
struct node *head = NULL;

/* 新しいノードを作る */
struct node *make_node(int data)
{
    struct node *p;

    p = (struct node *)
        malloc(sizeof(              ));
    if (p == NULL) exit(1);


    p->next = NULL;
    return p;
}

/* リストの先頭にノードを加える */
void push(int data)
{
    struct node *p;

    p = make_node(data);
    /* p の後ろに head をつなげる */


    head = p;
}
```

```
/* リストの先頭からノードを取り出す */
int pop(void)
{
    int data;
    struct node *p = head;

    /* データがあるかチェック */
    if (          ) {

        printf("Stack is empty.¥n");
        exit(1);
    }
    /* data を取り出し,新しい head を設定 */




    free(p);
    return data;
}

void print_stack(void)
{
    struct node *p;

    printf("[ ");
    for (p = head; p != NULL; p = p->next)


    printf("]¥n");
}

int main(void)
{
    /* 1.の main と全く同じ内容 */
}
```

4. 連結リストからデータを探索する関数を作成し，計算量を考察せよ（これも線形探索の一種である)。

```
struct node *search_list(int data)
{




}
```