Uvažte následující rekurzivní program:

```
static void Hanoi(char s, char t, char m, int c) {
  if (c == 0)
    return;
  else {
    hanoi(s, m, t, c-1);
    Console.WriteLine(s + "->" + t + ", ");
    hanoi(m, t, s, c-1);
  }
}
```

Následující program představuje neúplnou nerekurzivní variantu tohoto programu. Doplňte chybějící řádky.

```
class Task {
    public char s, t, m;
    public int c, segment;
    public Task(char s, char t, char m, int c) {
        __________;
        this.t = t;
        this.m = m;
        this.c = c;
        segment = 0;
    }
}
static void HanoiNR(char s, char t, char m, int c) {
    Stack<Task> tasks = new Stack<Task>();
    tasks.Push(new Task(s, t, m, c));
    while(tasks.Count>0) {
        Task task = tasks.Peek();
        switch (task.segment) {
           case 0:
              if (task.c == 0)
              {
                  tasks.Pop();
                  break;
              }
              ____________________________________________________________;
              task.segment += 1;
              break;
           case 1:
              Console.WriteLine(task.s + "->" + task.t + ", ");
              tasks.Push(new Task(task.m, task.t, task.s, task.c-
               1));
              ____________________________________________________________;
              break;
           case 2:
              tasks.Pop();
              break;
        }
    }
}
```

Do následující neúplné implementace ADT Zásobník doplňte metodu `Add(...)` pro přidání prvku. Řešení musí ošetřovat **všechny** situace, které mohou nastat.

```
class ArrayStack{
  int[] data;

  int freeIndex;

  public int Get() {
    if (freeIndex>0)
      return data[freeIndex-1];
    else throw new Exception();
  }

  void RemoveLast() {
    if (freeIndex>0)
      freeIndex--;
    else throw new Exception();
  }

  void Add
```

Do následující neúplné implementace ADT Zásobník doplňte metodu `add(...)` pro přidání prvku.

```
class Link{
  double data;
  Link next;
}

class LinkStack : IStack{
  public Link top;

  public double Get(){
    if (first!=null)
      return top.data;
    else throw new Exception();
  }

  void RemoveLast(){
    if (top!=null)
      top = top.next;
    else throw new Exception();
  }
```

Třída Stack implementuje zásobník hodnot typu int s následujícím rozhraním

```
interface IStack {
  void Add(int i); // prida
  int Get(); // vybere (neodebere)
  void Remove(); // odebere
}
```

Co vypíše následující program?

```
IStack stack = new Stack();
for (int i = 0;i<4;i++) {
  stack.Add(i*4);
  stack.Add(i/2);
  stack.Add(i+3);
}
for (int i = 0;i<5;i++) {
  Console.WriteLine(stack.Get());
  stack.Remove();
  stack.Remove();
  Console.WriteLine(stack.Get());
}
```

Třída `Stack` implementuje zásobník hodnot typu `int` s následujícím rozhraním

```
interface IStack {
  void Add(int i);
  int Get();
  void RemoveLast();
}
```

Co vypíše následující program?

```
IStack stack = new Stack();
for (int i = 0;i<10;i++) {
  stack.Add(i+3);
  stack.Add(i/2);
  stack.RemoveLast();
}
for (int i = 0;i<5;i++) {
  Cosnole.WriteLine(stack.Get());
  stack.RemoveLast();
}
```

Do následující neúplné implementace ADT Zásobník doplňte metodu `remove()` pro odebrání prvku.

```
class Link{
  double data;
  Link next;
}

class LinkStack : IStack{

  Link top;

  public double Get(){
    if (top!=null)
      return top.data;
    else throw new Exception();
  }

  void Add(double x) {
    Link l = new Link();
    l.data = x;
    l.next = top;
    top = l;
  }
```