UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Avaliação: **Avaliação 1** Data: / / Curso: _____

Disciplina: **Estrutura de Dados** Professor: **Adriano Rivolli**

Nome: _____ R.A.: _____ Nota _____

Obs: Não serão aceitas questões parcialmente corretas, em termos de lógica, apenas de sintaxe.

1) (1.5) Considerando o código a seguir relativo as operações padrões de pilha e fila, Informe a sequência numérica que deverá ser impresso na tela.

```
int main() {
  int i;
  stack s;
  queue q;

  newStack(&s);
  newQueue(&q);

  for (i=1; i<5; i++) {
    if (i%2) {
      push(&s, i);
    } else {
      enqueue(&q, i*2);
    }
  }

  push(&s, dequeue(&q));
  enqueue(&q, pop(&s));

  while(!isEmpty(&q)) {
    printf(" %d %d", pop(&s), dequeue(&q));
  }
}
```

3 8 1 4

2) (1.5) Escreva a função POP para a implementação de uma pilha em um vetor estático compartilhado, conforme a estrutura de dados a seguir. Considere que a propriedade base aponta para a primeira posição disponível do vetor e a propriedade top aponta para a próxima posição a ser inserida no vetor. Caso necessite de uma função adicional, você também deverá implementá-la. Retorne -1 caso a pilha esteja vazia.

```
typedef struct {
   int base;
   int top;
} stack;

int pop(int pilha[N], stack *s);
```

```
int pop(int pilha[N], stack *s) {
   if (s->top == s->base) {
     return -1;
   }
   s->top--;
   return pilha[s->top];
}
```

3) (1.5) Escreva o método ENQUEUE de uma fila utilizando a estrutura dinâmica a seguir. Considere que se head é nulo a fila está vazia.

```
typedef struct NodeList {
  char info;
  struct NodeList *prox;
} node;
```

```
typedef struct {
  node *head;
  node *tail;
} queue;
```

```
void enqueue(queue *f, node *n);
```

```
void enqueue(queue *f, node *n) {
  if (f->head == NULL) {
    f->head = f->tail = n;
  } else {
    f->tail->prox = n;
    f->tail = n;
  }
}
```

4) (1.5) Escreva o método para retornar um elemento em uma lista simplesmente encadeada SEM CABEÇA. Considere a estrutura 'node' apresentada na atividade anterior. Caso o valor informado não esteja presente na lista, retorne o valor nulo.

```
node *findNode(node *list, char value);
```

```
node *findNode(node *list, char value) {
  while(list != NULL && list->info != value) {
    list = list->prox;
  }
  return list;
}
```

5) (2.0) Considerando uma lista duplamente encadeada de valores inteiros, implemente uma função que receba como parâmetro uma lista duplamente encadeada e um valor inteiro e divida a lista em duas, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência do valor na lista original. A função deve retornar um ponteiro para a segunda sub-divisão da lista original. Considere que a função findNode já está implementada e que ela retorna NULL quando o valor pesquisado não for encontrado.

```
node* separa(node* L, int value);
```

```
node *separa(node *L, int value) {
  node *n, *p = NULL;
  n = findNode(L, value);
  if (n != NULL) {
    p = n->prox;
    n->prox = NULL;
    if (p != NULL) {
      p->ant = NULL;
    }
  }
  return p;
}
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

6) (2.0) Dada uma fila de inteiros formada por uma sequência de números positivos, construa uma função que remova da fila os valores pares e mantenha os ímpares na sua respectiva ordem. Exemplos:

2 4 5 6 7 9 ==> 5 7 9
1 4 2 5 8 7 ==> 1 5 7

Considere que as seguintes funções já estão implementadas:

```
void newQueue(queue *q);
void enqueue(queue *q, int value);
int dequeue(queue *q);
int isEmpty(queue *q);
```

```
void removePar(queue *q) {
  int aux;

  enqueue(q, 0);
  aux = dequeue(q);

  while(aux > 0) {
    if (aux % 2) {
      enqueue(q, aux);
    }
    aux = dequeue(q);
  }
}
```

Boa prova!!!