Pawelec:

zadanie 1.

1. Zaimplementuj listę korzystając z dynamicznej alokacji pamięci.

2. Węzeł listy ma być zdefiniowany jako:

```
typedef struct List
{
        int value;
        struct List* next;
}List;
```

3. Maksymalna ilość elementów na liście ma być konfigurowalna przez makro MAX_ELEMS.

4. Zaimplementuj metody dodawania/usuwania elementów na listę:

void push_back()
• zwracana wartość to wartość pola value z węzła listy, jeżeli ilość elementów przekrowczyła MAX_ELEMS wówczas jest zwracana wartość -1

int pop_front()
• jeżeli lista jest pusta, zwracana wartość ma wynosić -1

5. Stwórz pule N-wątków z czego:
• co najmniej dwa wątki mają zapisywać dane
• co najmniej dwa wątki mają odczytywać dane

6. Zidentyfikuj sekcje krytyczne w programie i odpowiednio je zabezpiecz.

7. Jeżeli lista jest pusta, wówczas czytające wątki mają być uśpione.

8. Jeżeli lista jest pełna, wówczas zapisujące wątki mają być uśpione.

//

1. Zaimplementuj stos przy wykorzystaniu tablicy o rozmiarze N.

2. Zaimplementuj metody dodawania/usuwania do/z stosu:

void push_pop()
• jeżeli stos jest pełny, wówczas zwracana wartość ma wynosić -1

int pop_front()
• jeżeli stos jest pusty, wówczas zwracana wartość ma wynosić -1

3. Instancja tego stosu ma znajdować się w pamięci współdzielonej dla dwóch powiązanych procesów (producent, konsument).

4. Zidentyfikuj sekcje krytyczne w programie i odpowiednio je zabezpiecz.

5. Jeżeli stos jest pusty, wówczas czytający proces (konsument) ma być uśpiony.

6. Jeżeli stos jest pełny, wówczas zapisujący proces (producent) ma być uśpiony.