Program 8.

Implement a Buddy Allocator System:

Code:

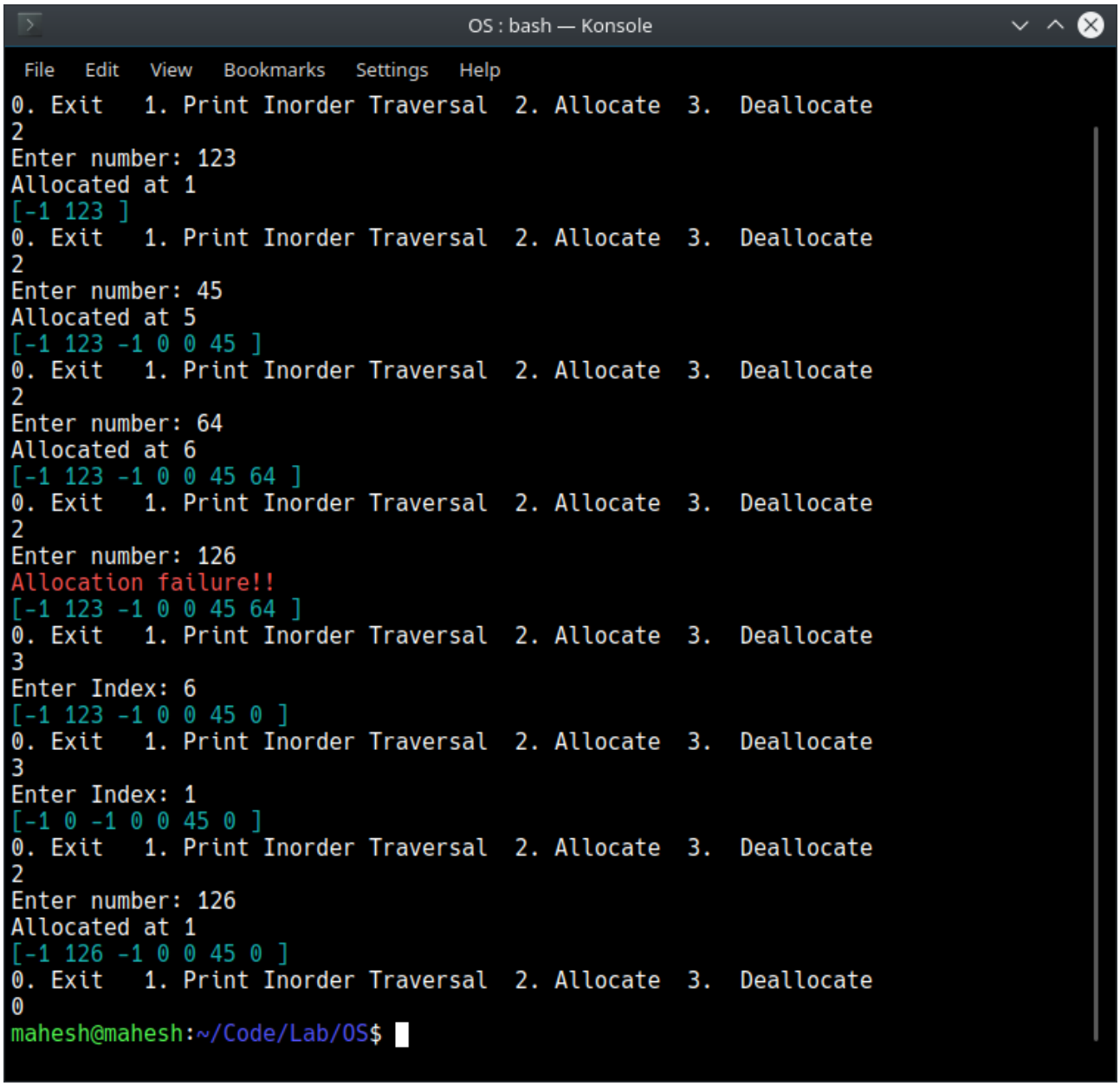
```
#include<stdio.h>
// 0 -> unallocated
// positive integer -> allocated
// -1: Some direct or indirect child is allocated

int tree[1024] = {0}, len = 0;

const int total_size = 256;

int parent(int i) {
      return (i-1)/2;
}

int left(int i) {
      return 2*i + 1;
}

int right(int i) {
      return 2*i+2;
}

int sibling(int i) {
      if (i == 0) return 0;
      return (i % 2 == 0) ? i - 1 : i + 1;
}

// An allocation is index into this array;
// There we store number of bytes allocated
// Or -1 if a child is allocated
```

```
// 0 if untouched (default)

int isfit(int request, int block) {
	return request <= block && request > block/2;
}

// O(log n) recursion

int alloc(int request, int begin, int size) {
	int lres, res;
	// OOM or bad request
	if (request > size || request < 1) return -1;

	if (len < begin+1) len = begin+1; // remember the maximum value of a child
ever referenced

	// Already allocated
	if (tree[begin] > 0 ||
			(isfit(request, size) && tree[begin] == -1)) return -1;

	// if fit & unallocated, allocate
	if (isfit(request, size)) {
		tree[begin] = request;
		return begin;
	}

	// split the memory and check with both children
	lres = alloc(request, left(begin), size/2);
	res = (lres == - 1) ? alloc(request, right(begin), size/2) : lres;
	if (res != -1) {
		tree[begin] = -1;
	}
	return res;
}

void delete(int index) {
	int size;
```

```
	// unset the allocation
	// coalesce if neighbour is also 0, and free parent then.
	// Max O(log n) depth recursion, should be safe even if not tail-call optimized
	if(tree[index] == 0) {
		printf("\e[31;1mDOUBLE FREE OR WILD PTR!!\e[0m\n");
		return;
	}

	size = tree[index];
	tree[index] = 0;
	if (index != 0 && tree[sibling(index)] == 0) {
		delete(parent(index));
	}

	return;
}

void print_heap(int begin) {
	if (begin >= len) return;
	print_heap(left(begin));
	printf("%d ", tree[begin]);
	print_heap(right(begin));
}

void print_array() {
	printf("\e[36m[");
	for(int i = 0; i < len; i++) {
		printf("%d ", tree[i]);
	}
	printf("]\e[0m\n");
}

int main() {
	int ch, n, r;
	while(1) {
```

```
            printf("0. Exit   1. Print Inorder Traversal  2. Allocate  3.
Deallocate\n");
            scanf("%d", &n);
            switch(n) {
            case 0: return 0;
            case 1: printf("\e[33m Inorder traversal of tree: ");
                  print_heap(0);
                  printf("\e[0m\n");
                  break;
            case 2: printf("Enter number: ");
                  scanf("%d", &n);
                  r = alloc(n, 0, 256);
                  if (r == -1) {
                        printf("\e[31;1mAllocation failure!!\e[0m\n");
                  } else {
                        printf("Allocated at %d\n", r);
                  }
                  break;
            case 3: printf("Enter Index: ");
                  scanf("%d", &n);
                  if (tree[n] == -1) {
                        printf("\e31;1mNot a valid allocation!!\e[0m\n");
                  } else {
                        delete(n);
                  }
                  break;
            default: printf("Invalid Option!!\n");
                   break;
            }
            print_array();
      }
      return 0;
}
```

Output:


OS : bash — Konsole
File Edit View Bookmarks Settings Help
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
2
Enter number: 123
Allocated at 1
[-1 123]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
2
Enter number: 45
Allocated at 5
[-1 123 -1 0 0 45]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
2
Enter number: 64
Allocated at 6
[-1 123 -1 0 0 45 64]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
2
Enter number: 126
Allocation failure!!
[-1 123 -1 0 0 45 64]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
3
Enter Index: 6
[-1 123 -1 0 0 45 0]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
3
Enter Index: 1
[-1 0 -1 0 0 45 0]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
2
Enter number: 126
Allocated at 1
[-1 126 -1 0 0 45 0]
0. Exit 1. Print Inorder Traversal 2. Allocate 3. Deallocate
0
mahesh@mahesh:~/Code/Lab/OS$